برمجة الكمبيوتر

بلغة التجميع

IBM PC COMPATABLES
ASSEMBLY LANGUAGE

تأليف

د. زياد القاضي - د. سامي سرحان - د. عبد الفتاح سلمان

م. ابراهيم غريب - م. رشاد رصرص

دار المستقبل للنشر والتوزيع

عمان - الاردن

ص.ب ١٨٤٢٤٨ - هاتف ٦٣٦٣٢٧

الطبعة الاولى

رقم الاجازة المتسلسل: ٣٩١/ ٧/ ١٩٩٠
رقم الايداع لدى دائرة المكتبات والوثائق الوطنية (٤٣٩/ ٧/ ١٩٩٠).
تاريخ تقديم الطلب: ١٠/ ٧/ ١٩٩٠

دار المستقبل للنشر والتوزيع
عمان ـ الاردن
ص.ب ١٨٤٢٤٨ ـ هاتف ٦٣٦٣٢٧

بسم الله الرحمن الرحيم

«أَلَمْ تَرَوْا أَنَّ اللَّهَ سَخَّرَ لَكُم مَّا فِي السَّمَاوَاتِ ومَا فِي الأَرْضِ
وأَسْبَغَ عَلَيْكُم نِعمَهُ ظَاهِرَةً وبَاطِنَةً»

صدق الله العظيم

الاهداء

إلى كل عربي نذر نفسه للعطاء

إلى كل من ساهم ويساهم في رفع راية امتنا

إلى كل الاعين الساهرة على خدمة الامة العربية

نهدي كتابنا هذا ليساهم في المكتبة العربية بحقل جديد

المؤلفون

المحتويات

المقدمة

منذ سنوات عدة أصبح الميكروكمبيوتر حقيقة واقعة وملموسة يتعامل معه وبه جميع افراد المجتمع وعلى كافة المستويات العلمية والتخصصية المختلفة .

فقد دخل الميكروكمبيوتر جميع المجالات الخاصة والعامة من مؤسسات وجامعات ومعاهد ومدارس ومكاتب علمية وهندسية وتجارية وكذلك دخل الى المنازل واستعمله الافراد لاغراضهم الشخصية .

ونظرا لهذا الانتشار الواسع لاجهزة الكمبيوتر الميكروية جاء تقديمنا لهذا الكتاب اسهاماً منا باثراء المكتبة العربية والتي هي بأمس الحاجة الى الكتب العلمية والمكتوبة باللغة العربية املين بذلك تسليح القارىء العربي بالمهارات اللازمة لاستخدام الميكروكمبيوتر.هذا وقد استعرضنا في هذا الكتاب الاسس والمفاهيم الاساسية اللازمة لفهم ومعرفة لغة التجميع وكيفية استخدامها وقد راعينا هنا التركيز على التعليمات الخاصة بالمعالج الدقيق INTEL 8086 نظرا لما يمتلكه هذا المعالج من مزايا وقدرات فائقة ولاستخدامه في كافة اجهزة IBM والاجهزة المتوافقة مع هذه الاجهزة .

يتضمن الكتاب شرح موسع لكافة العمليات والتعليمات الخاصة بلغة التجميع اضافة الى استخدام لغة التجميع لبرمجة الادخال والاخراج ومعالجة الملفات .

وقد تضمنت الوحدات المختلفة الكثير من التطبيقات العملية أملاً منا في تسهيل ايصال المعلومات الى القراء ولتعزيز المعلومات النظرية المطروحة وذلك لتوسيع مجال الاستفادة من هذه المفاهيم العلمية وليسهل استخدامها من قبل كافة القطاعات التخصصية كهندسة الكمبيوتر وعلومه اضافة الى التخصصات الاخرى والى كل مهتم بعلم الكمبيوتر .

وفي الختام نسأل الله ان نكون قد وفقنا لخدمة القارىء العربي املين ان يتحقق لديه اكبر قدر من الفائدة العلمية ، داعين الله ان يزيد من بصيرته ، ويرزقه من واسع علمه ويرفعه درجة فوق درجاته .

المؤلفون
عمان – ١٩٩٠

الوحدة الاولى

تركيب الميكروكمبيوتر ولغة التجميع

- تركيب الميكروكمبيوتر
- المقاطع
- المسجلات
- تركيب لغة التجميع
- هيكل برنامج التجميع
- تصنيف التعليمات
- اشارات العمليات

تركيب الميكروكمبيوتر

يعتبر المعالج الميكروي الوحدة الوظيفية الرئيسية في الميكروكمبيوتر ويمكن تجزئة هذه الوحدة الى قسمين : وحدة التنفيذ (execution unit) ووحدة المواجهة البينية-bus inter) (face unit حيث تقوم وحدة التنفيذ بتنفيذ كافة التعليمات بينما تقوم وحدة المواجهة بتزويد وحدة التنفيذ بالتعليمات والبيانات اللازمة لاجراء عملية المعالجة .

تتألف وحدة التنفيذ من وحدة الحساب والمنطق والتي تقوم باجراء العمليات الحسابية والمنطقية ، وحدة التحكم ومجموعة من المسجلات الخاصة . اما وحدة المواجهة البينية فتتألف من اجزاء رئيسية هي : ضابط التحكم بالناقل ، دور التعليمات ومسجلات المقاطع وتقوم هذه الوحدة باجراء الوظائف التالية :

– الاشراف على نقل البيانات والتعليمات الى وحدة التنفيذ والذاكرة ووحدات الادخال والاخراج .

– عنونة الذاكرة باستخدام مسجلات المقاطع .

– تحضير التعليمات اللازمة وجلبها من الذاكرة وذلك من خلال تنظيم دور التعليمات حيث تقوم الوحدة التنفيذية بأخذ الاوامر والتعليمات من هذا الدور بدلا من الرجوع الى الذاكرة مما يعني زيادة سرعة التنفيذ .

تعمل وحدة التنفيذ ووحدة المواجهة البينية بشكل متوازٍ حيث تقوم وحدة التنفيذ باجراء عمليات المعالجة بينما تقوم وحدة المواجهة بتجهيز البيانات والتعليمات اللازمة .

اما الوحدة الوظيفية الثانية فهي الذاكرة الرئيسية حيث تتألف هذه الذاكرة من جزئين :

– ذاكرة القراءة فقط(Read Only Memory) وتستخدم لتخزين برامج بدء التشغيل ، برامج الفحص والتحميل في ذاكرة القراءة والكتابة .

– ذاكرة القراءة والكتابة (Random Access Memory) وهي الجزء الفعال والمستخدم في تنفيذ برامج المستخدم حيث تستخدم لتخزين التعليمات والبيانات اللازمة للمعالجة . وعند الحديث عن الذاكرة يقصد بذلك ذاكرة (RAM)

تقسم الذاكرة الى مواقع بحيث يحمل كل موقع عنوانا ويتعامل المعالج مع البيانات المخزنة في المواقع بالبايت او الكلمات ويراعى عند الرجوع للبيانات المخزنة في الذاكرة انها قد تكون مكتوبة بشكل عكسي (Reverse) خاصة عندما يكون طول البيانات ١٦ خلية ثنائية حيث تظهر محتويات الخانات من صفر الى ٧ في الموقع الاول اما الخانات من ٨ – ١٥ فتظهر في الموقع التالي وبهذا تظهر البيانات وكأنها مقلوبة فمثلا الرقم (0506H)

يخزن في الذاكرة بشكل مقلوب (0605)

يخصص جزء من الذاكرة الى المقاطع اللازمة حيث يخصص ٦٤ كيلوبايت لكل مقطع والشكل التالي يوضح اهم تقسيمات الذاكرة .

Address	Area
	256K of RAM on board
256 K	3884 K RAM memory expansion in I/0 channel
640 K	128 K graphic / display video buffer (RAM)
768 K	192 K memory expansion area (ROM)
960 K	64 K base system (ROM)
1024 K	

المقاطع (Segments)

يعرف المقطع على انه جزء من الذاكرة وقد يصل حجمه الى ٦٤ كيلو بايت وتقسم المقاطع الى عدة اقسام :

١ - مقطع التعليمات (Code segment) حيث يخصص هذا المقطع لتخزين تعليمات الالة المراد تنفيذها حيث تخزن اول تعليمة في بداية هذا المقطع وتتم عنونة هذا المقطع بواسطة مسجل مقطع التعليمات (CSR) .

٢ - مقطع البيانات (Data segment) حيث يحتوي هذا المقطع على البيانات المعرفة والثوابت ومنطقة العمل اللازمة لتنفيذ البرنامج ويعنون هذا المقطع باستخدام مسجل مقطع البيانات (DSR)

٣ - مقطع الحزمة (Stack segment) ويحتوي هذا المقطع على عناوين الرجوع من البرنامج الى نظام التشغيل وعناوين الرجوع عند الانتهاء من تنفيذ البرامج الفرعية الى التعليمات التي قطع عندها التنفيذ باستخدام تعليمات الاستدعاء (CALL) وتتم عنونة

هذا المقطع بواسطة مسجل مقطع الحزمة (SSR)

٤ – المقطع الاضافي (Extra segment) ويستخدم هذا المقطع لاغراض خاصة ، وتتم عنونة هذا المقطع بواسطة مسجل المقطع الاضافي (ESR)

تستخدم المسجلات المخصصة بالمقاطع لتحديد بداية المقطع ولايجاد العنوان الفعلي يتم اضافة القيمة المخزنة في المسجل الى عنوان البيانات او التعليمات المخزنة في المقطع .

المسجلات REGISTERS

يضم المعالج الميكروي (INTEL 8086) مجموعة من المسجلات تستخدم للتحكم في تنفيذ التعليمات وعنونة الذاكرة بالاضافة الى استخدامها في تنفيذ العمليات الحسابية ، ويتألف كل مسجل من ١٦ خلية ثنائية ومن اهم هذه المسجلات :

١ – مسجلات المقاطع (Segment registers)

تستخدم هذه المسجلات لعنونة مقاطع الذاكرة المستخدمة في البرنامج ومن اهم هذه المسجلات .

أ – مسجل مقطع التعليمات (CSR) حيث يستخدم هذا المسجل لعنونة مقطع التعليمات بحيث يخزن به العنوان الابتدائي للمقطع .

ب – مسجل مقطع البيانات (DSR) ويستخدم لتخزين العنوان الابتدائي لمقطع البيانات .

جـ – مسجل مقطع الحزمة (SSR) ويخزن به العنوان الابتدائي لمقطع الحزمة .

د – مسجل المقطع الاضافي (ESR) ويخزن به عنوان المقطع الاضافي (ان استخدم) .

٢ – مسجلات الاستخدام العام (General purpose registers)

تستخدم هذه المسجلات لاغراض عامة في البرنامج حيث يمكن استخدامها في العمليات الحسابية والمنطقية ويمكن لهذه المسجلات التعامل مع البيانات بطول ٨ بت (byte) أو بطول ٢ بايت ومن اهم هذه المسجلات :

أ – المسجل AX (المركم : accumulator) ويستخدم في عمليات الادخال والاخراج ، العمليات الحسابية والمنطقية ويمكن التعامل مع هذا المسجل وذلك باجراء كافة العمليات على محتوياته والمؤلفة من ١٦ خلية ثنائية كما ويمكن تجزئته الى قسمين كل منهما بطول ٨ خلايا ثنائية والتعامل مع كل قسم على حده

AX|AH|AL|

على اعتبار ان الخلايا الثنائية من صفر الى ٧ موجودة في AL اما الخلايا من ٨ - ١٥ فموجودة في AH

ب — المسجل BX (base register) حيث يستخدم لفهرسة الذاكرة ويمكن التعامل مع اقسامه .

BX : /BH//BL/

جـ - المسجل CX (Counter) ويستخدم كعداد للتحكم بعدد مرات التكرار والازاحة ويقسم الى قسمين

CX : /CH/CL/

د - المسجل DX (Data register) حيث يستخدم في بعض عمليات الادخال والاخراج والعمليات الحسابية ويمكن تجزئة هذا المسجل الى

DX :/DH/DL/

٣ - **مسجلات التأشير (pointer registers)**

وتضم هذه المسجلات :

أ - مؤشر الحزمة (SP) يرتبط هذا المسجل بالحزمة ويستخدم لعنونتها .

ب - مؤشر القاعدة (BP base pointer) ويستخدم للاشارة الى العوامل (البيانات أو العناوين الداخله او الخارجه من الحزمة) .

٤ - **مسجلات الفهرسة (Index registers)**

حيث تستخدم هذه المسجلات في العنونة الموسعة وفي عمليات الجمع والطرح ومن هذه المسجلات :

أ - المسجل SI (Source Index) يستخدم في معالجة السلاسل وعادة ما يرتبط بالمسجل DSR .

ب - المسجل DI (Destination Index) ويستخدم ايضا في بعض العمليات المتعلقة بمعالجة السلاسل ويرتبط عادة بالمسجل ESR .

جـ - مسجل مؤشر التعليمة (Instruction Pointer IP) يستخدم هذا المسجل اثناء تنفيذ التعليمات حيث يخزن به مقدار الازاحة في العنوان وذلك لحساب العنوان الحقيقي للتعليمة.

٥ - **مسجل الحالة (Flag Register F)** تستخدم خلايا هذا المسجل لاظهار حالة المعالج بعد تنفيذ تعليمة معينة وقد يؤدي تنفيذ بعض التعليمات الى تغير في حالة المعالج والتي يتم الاحتفاظ بها في هذا المسجل ومن اهم هذه الحالات.

أ - حالة الفائض (Overflow "O") والتي تعني وجود حمل بعد اعلى خانة من اليسار .

ب - حالة تحديد الاتجاه (Direction "D") حيث يُحدد اتجاه عمليات النقل في السلاسل (يسارا او يمينا) او اتجاه عملية مقارنة السلاسل (في الحالات التي يزيد فيها طول السلسلة عن كلمة) .

جـ - حالة الاعتراض (Interrupt "I") ويدل محتوى هذه الخلية على امكانية تقبل الاعتراض او عدمها .

د - حالة المتابعة (Trap : "T") يمكن من خلال استخدام الخلية المخصصة وجعل قيمتها "1" متابعة تنفيذ كل تعليمة على حده (Step - by - step) .

هـ - حالة الصفر (Zero "Z") تشير خانة الصفر الى نتيجة العملية الحسابية ("1" يشير الى الصفر "0" يشير الى نتيجة غير صفرية).

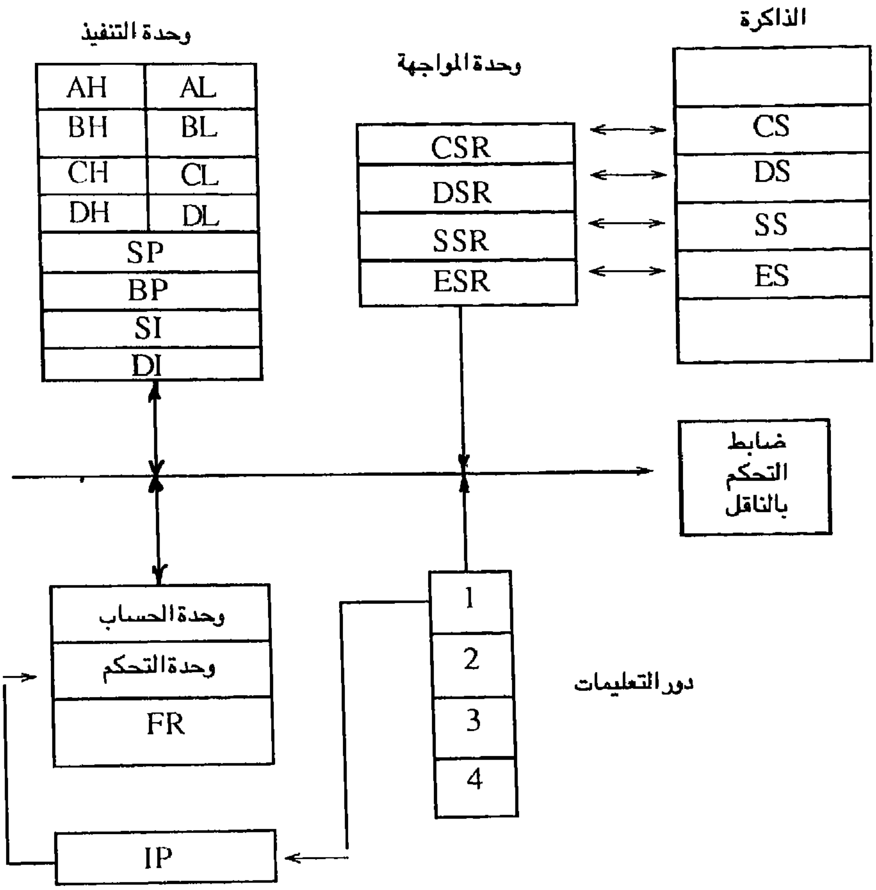
و - الاشارة (Sign "s") يتم اظهار اشارة النتيجة هل هي سالبة ام موجبة اعتمادا على قيمة الخانة المخصصة للاشارة. (S=0 موجب)

ز - الفائض الخارجي (Auxiliary Carry "A") لاظهار الفائض الناتج بعد الخانة رقم "٣" في البيانات المؤلفة من ٨ خلايا ثنائية .

ح - التحقق (Parity "P") للدلالة على نوع التحقق المستخدم هل هو زوجي او فردي. (P=1 زوجي)

ط - الحمل (Carry "C") تحتوي الخلية المخصصة على الحمل الناتج بعد اعلى خانة عند اجراء عملية حسابية او ازاحة او دوران .

والشكل التالي يوضح ترابط المعدات الاساسية في المعالج



تركيب لغة التجميع

- البرنامج المصدري في لغة التجميع .

يتكون برنامج لغة التجميع من سلسلة من الجمل مرتبة ترتيبا منطقيا وهذه الجمل يمكن ان تكون اوامر للمعالج تطلب منه تنفيذ عمليات معينة او تكون توجيهات للمترجم تبين له ما يجب عمله اثناء ترجمة البرنامج المصدري وهذه التوجيهات فعالة فقط أثناء ترجمة البرنامج المصدري فقط .

سوف نقوم بدراسة تعليمات 8086 ، والتوجيهات المستخدمة في المعالج بالتفصيل.

١ - تعليمات 8086

كما ذكرنا سابقا فان التعليمات هي بمثابة اوامر للمعالج لتنفيذ عملية معينة والشكل

العام للتعليمة في برنامج لغة التجميع هو كما يلي

[label:] operation [operands], comments

جميع حقول التعليمة هي اختيارية ما عدا حقل operation ، والحقول الاختيارية هي التي يمكن ان تتواجد في بعض التعليمات وتختفي في الاخرى حسب الحاجة لها . وتدل الاقواس [ ] على تلك الحقول.

الحقل label هو حقل العلامة ، ويكون على شكل اسم رمزي ، ويأتي هذا الاسم في بداية التعليمة عند الحاجة الى نقل التنفيذ اليها خلال تنفيذ البرنامج.

واختيار اسم العلامة يجب ان يخضع للشروط التالية وهي شروط تكوين الاسماء في لغة التجميع 8086 . وهي

١) يتكون الاسم من الحروف A - Z

٢) الارقام من 0 حتى9

٣) رموز خاصة @ , - , $, . , . , ?, ويجب ان لا يكون ذلك الاسم من الرموز المحجوزة في اللغة مثل اسماء المسجلات المختلفة .

حقل operation وهي الاختصار الرمزي لاسم العملية المطلوب تنفيذها من قبل المعالج ، ويمكن ان يتكون من رمزين حتى ٦ رموز.

حقل المعاملات operands – وهي المعاملات المطلوب اجراء العمليات عليها – ويمكن ان يحتوي حقل التعليمة على معاملين او معامل ، ويمكن ان لا يحتوي عليه ، اذا وجد في حقل المعامل معاملان توضع فاصلة بينهما .

ويسمى المعامل الاول بالمعامل المصدري ، والمعامل الثاني بالمعامل المستقبل

مثال

Mov AX ,BX

حقل الملاحظات وهو حقل اختياري في جميع التعليمات ويستخدم لوصف العملية ، او اعطاء ملاحظات مختلفة

مثال

Mov CX,9 ; transfer 9 to CX

– **التوجيهات** : – وهي تبين للمترجم ما يجب عليه عمله اثناء ترجمة البرنامج المصدري ، فهي موجهة للمترجم فقط وليس للمعالج .

الصيغة العامة للتوجيه

(Name) Directive [operand] [; comment]

كما هو مبين نجد ان جميع الحقول اختيارية ما عدا حقل واحد وهو حقل Directive

وتصنف التوجيهات الى خمس مجموعات وهي

١ – توجيهات البيانات
٢ – توجيهات القطاعات والاجراءات
٣ – توجيهات وصف التكتلات
٤ – توجيهات التحكم
٥ – توجيهات الاتصال بين البرامج

١ – توجيهات البيانات : DATA DIRECTIVES

تقوم هذه المجموعة بتعريف الثوابت والمتغيرات المستخدمة في البرنامج والجدول يوضح تلك التوجيهات ، واشكالها العامة.

مثال X EQU – 200

باستخدام التوجيه EQU ثم تعريف الموقع X كثابت يحتوي على رقم 200 واكبر قيمة يمكن ان يأخذها في هذه الحالة تكون مكونة من ١٦ خانة ثنائية

اسم التوجيه	الشكل العام
DB	[Name] DB expr [, ...]
DW	[Name] DW expr [, ...]
DD	[Name] DD expr [, ...]
EQU	[Name] EQU expr [,...]

expr يمكن ان يكون ثابت ، او متغير ، او مصفوفه

امثلة على ذلك

الموقع × يحتوي على ثابت –	X DB 20
متغير × يحتوي على عنوان y	X DW Y
مصفوفه	X DB 1,2,5,10

٢ – توجيهات القطاعات : يبين الجدول تلك التوجيهات واشكالها العامة . التوجيه SEGMENT و ENDS – تقوم كلاهما بتقسيم البرنامج الى قطاعات ، حيث ان SEGMENT تشير الى بداية القطاع و ENDS تشير الى نهاية القطاع . كل برنامج مصدري يمكن ان يحتوي على اربع مقاطع كل منها يبدأ بـ SEGMENT وينتهي بـ ENDS .

التوجيه ASSUME – يقوم باخبار المترجم عن مكان وجود عنوان بداية كل قطاع في البرنامج ، والذي يمكن ان يكون في احدى مسجلات القطاعات .
مثال:

ASSUME CS : COD , : stack, DS: DAT .

نفهم من هذا المثال ان عنوان بداية قطاع COD موجود في مسجل القطاع CS ، وعنوان بداية قطاع DAT موجود في مسجل القطاع DS

اسم التوجيه	الشكل العام
SEGMENT	seg-name SEGMENT
ASSUME	ASSUME seg-reg : seg-name
PROC	name PROC

```
name    PROC
         .
         .
         .
        RET
name    ENDP
```

التوجيه PROC – يبين للمترجم بداية برنامج فرعي ، ينتهي هذا البرنامج الفرعي بالتوجيه ENDP . والشكل العام للبرنامج المبين في الجدول . اذا احتوى البرنامج الفرعي على الكلمة المحجوزة FAR بعد PROC فهذا يعني بأنه يمكن استدعاء هذا البرنامج الفرعي من اي قطاع من القطاعات اما اذا احتوى على NEAR فاستدعاؤه يمكن فقط من داخل نفس القطاع الموجود فيه .

٣ – توجيهات التكتل

التوجيه GROUP – يجمع مجموعة من القطاعات في منطقة ذاكرة لا تتجاوز سعتها 64 KB ، مع اعطاء هذه المجموعة اسم واحد.

مثال : اذا كان لدينا قطاع البيانات في البرنامج تحت اسم DAT وقطاع التعليمات تحت اسم COD فانه يمكن تجميعها تحت اسم واحد مثلا XX بالتوجيه التالي

XX GROUP DAT, COD

٤ – توجيهات التحكم

التوجيه ORG – اخبار المترجم عن عنوان بداية منطقة الذاكرة المخصصة لتخزين البرنامج الهدفي ، الناتج من الترجمة

الشكل العام

ORG exp.

حيث exp . هي عنوان منطقة التخزين.

ORG 100H

تعمل على تخزين البرنامج من عنوان 100H

- التوجيهEND - اخبار المترجم عن نهاية البرنامج المصدري . وهي تأتي في نهاية البرنامج.

٥ - توجيهات ربط البرنامج مع برامج اخرى

التوجيه PUBLIC - تقوم بالاعلان عن المتغيرات المشتركة بين البرامج المطلوب ربطها مع بعضها البعض ، وجعلها معرفة لكل منها . باختصار تعريف المتغيرات التي يمكن الرجوع اليها من برامج اخرى.

التوجيه EXTRN - تقوم بتعريف المتغيرات والرموز والتي تكون معرفة في برنامج اخر.

مثال : للوصول لموقع الذاكرة ZY من برنامجين مختلفين ، في البرنامج الذي يتم فيه تعريف ZY نكتب الجمل التالية

```
PUBLIC ZY
ZY DB ?
```

في البرنامج الاخر الذي سوف يستخدم ZY يتم التعريف التالي قبل الاستخدام

```
EXTRN ZY
```

هيكل برنامج التجميع

قبل البدء في وصف التعليمات الخاصة بلغة التجميع لا بد من القاء الضوء على المكونات الرئيسية للبرنامج المصدري والمكتوب بلغة التجميع . ومن هذه المكونات يمكن ابراز ما يلي :

١ - ترويسة البرنامج (Program heading)

٢ - مقطع الحزمة

٣ - مقطع البيانات

٤ - مقطع التعليمات

ترويسة البرنامج

تضم ترويسة البرنامج معلومات مفيدة قد تستخدم للدلالة على نوع البرنامج والمشكلة المراد حلها كما انها قد تضم معلومات تستخدم لاغراض طباعة البرنامج بشكل منظم وذلك

بتحديد عدد الاسطر المراد طباعتها في الصفحة الواحدة وعدد الرموز الواجب طباعتها في السطر الواحد اضافة الى امكانية تكرار ترويسة البرنامج في بداية كل صفحة.

وقد تتكون الترويسة من التعليمات التالية

– تعليمة تنظيم الصفحة (PAGE) حيث يحدد بواسطة هذه التعليمة عدد الاسطر المراد طباعتها في الصفحة الواحدة اضافة الى عدد الاحرف في السطر الواحد فمثلا لطباعة ٥٠ سطرا بطول ٨٠ حرف يمكن استخدام الامر التالي

PAGE 50,80

عدد الاسطر في الصفحة الواحدة يتراوح بين ١٠ و ٢٥٥ اما عدد الاحرف في السطر الواحدفيتراوح بين ٦٠ و ١٣٢ حرفا .

وتجدر الاشارة هنا ان وجود تعليمة الصفحة في البرنامج ليس ضروريا وعند اهمالها فان عدد الاسطر يحدد بالرقم ٦٦ اما عدد الاحرف في السطر فيحدد بالرقم ٨٠ .

عند استخدام هذا الامر فان عداد الصفحة يأخذ القيمة الدالة على عدد الاسطر ويبدأ العداد بالتناقص بمقدار ١ بعد طباعة كل سطر حتى تصبح قيمته مساوية للصفر عندها يتم قلب الصفحة الى الصفحة التالية وهكذا . وفي بعض الاحيان قد يتطلب الامر قلب الصفحة قبل وصول قيمة عداد الصفحة الصفر عندها يمكن استخدام الامر PAGE فقط وبدون تحديد عدد الاسطر والاحرف ومصادفة هذا الامر يُجبر الالة الطابعة على القفز الى بداية صفحة جديدة.

– موضوع البرنامج (TITLE) يتم تحديد موضوع البرنامج باستخدامTITLE حيث يُحدد اسم البرنامج ومختصر موجز عن طبيعة البرنامج كما يلي:

TITLE ASMPRT Assembler program to print

ويؤدي استخدام TITLE في البرنامج الى طباعة موضوع البرنامج في بداية كل صفحة ويعتبر استخدام الموضوع غير اجباري في البرنامج . هذا ويمكن استخدام مواضيع فرعية في البرنامج وذلك للدلالة على البرامج الفرعية حيث يمكن استخدام -SUB TITLE ليقوم بمفعولTITLE لكن فقط داخل البرنامج الفرعي .

توثيق البرنامج

تمتلك لغة التجميع القدرة على التوثيق والتي تعني سهولة مراجعة البرنامج وفهمه وسهولة متابعته وذلك باستخدام جمل غير تنفيذية تتضمن بعض الملاحظات اللازمة لتفسير التعليمات المستخدمة في البرنامج أو لتوضيح الاجراءات المنفذة في البرنامج .

تستخدم الملاحظة في اي جزء من البرنامج على ان تكون مسبوقة بالفاصلة المنقوطة كما يلي

; PROGRAM TO ADD TWO NUMBERS
MOV BX,25 ; MOVE 25 TO BX

المقاطع

يتكون برنامج التجميع من مقطعا واحدا على الاقل- مقطع التعليمات - وقد يحتوي البرنامج على مقطع البيانات والمخصص لتعريفها أو على مقطع الحزمة او المقطع الاضافي وعند استخدام هذه المقاطع في البرنامج لا بد من الاعلان عنها وذلك باعطاء كل مقطع اسما متبوعا بالتعليمة Segment ومجموعة من الخيارات (Options) كما يلي

```
segmentname    SEGGMENT  Options
                  :
                  :
segmentname    ENDS
```

ينطبق على اسم المقطع ما ينطبق على كافة الاسماء المستخدمة في البرنامج ومن اهم القواعد الواجب مراعاتها عند اختيار الاسماء ما يلي :

- استخدام الاحرف الانجليزية الكبيرة او الصغيرة .
- يمكن استخدام الارقام في الاسم .
- امكانية استخدام الاشارات الخاصة في الاسم .

- ان يبدأ الاسم بحرف او رمز خاص ولا يجوز ان يبدأ الاسم برقم .
- ان لا يزيد طول الاسم عن ٣١ رمزا .

يرتبط القاطع عادة بعنوان يخزن في احد المسجلات الخاصة بالمقاطع لذا تتم عملية ربط المقطع بالمسجلات المخصصة لها وذلك باستخدام الامر ASSUME كما يلي.

ASSUME SS :STACKNAME, DS : Data segment , CS : code segment

اما مجموعة الخيارات فتستخدم لتحديد بداية المقطع (PARA) او لتحديد امكانية ربط المقطع مع مقاطع اخرى (Combine) وسوف نستعرض هذه الامور في مواضيع لاحقة.

والشكل التالي يوضح المكونات الاساسية لبرنامج التجميع . لاحظ ان كل مقطع يجب ان يبدأ بالاسم وينتهي بالامر ENDS

```
                PAGE 50,60
;====== 50 LINES PER PAGE & 60 CHARACTERS PER LINE
TITLE    CHAPT1  ASSEMBLER  STRUCTURE
;========SEGMENT DEFINITION======================
;1)       STACK SEGMENT  ........................
;NAME    SEGMENT    OPTIONS
STACKSG SEGMENT    PARA STACK 'STACK'
;           ...
;           ...
;           ...
STACKSG ENDS
;
;2)        DATA SEGMENT   ........................
;================================================
DATASG   SEGMENT PARA 'DATA'
;FLDNAMES        DEFINITION
;--------        ----------
; ...       ...
; ....      ...
; ...       ...
DATASG   ENDS
;=================================================
;3)         CODE SEGMENT
;=================================================
CODESG   SEGMENT PARA 'CODE'
BEGIN    PROC FAR
                ASSUME  CS:CODESG,DS:DATASG,SS:STACKSG,ES:DATASG
;.......   SET
;.......   OF
;........ INSTRUCTIONS
BEGIN    ENDP
CODESG   ENDS
                END      BEGIN
```

تصنيف التعليمات

طاقم التعليمات في المعالج 8086 يحتوي على 92 نوع من التعليمات الاساسية، التي يمكن تصنيفها الى 7 مجموعات حسب الوظيفة التي تقوم بها كل مجموعة وهي

١ – تعليمات نقل البيانات – وهي التعليمات التي تقوم بتحريك البيانات بين مسجلات المعالج ومواقع الذاكرة وموانىء الادخال والاخراج .

٢ – التعليمات الحسابية : تقوم هذه التعليمات بالعمليات الحسابية المنفذة على الاعداد الثنائية – والاعداد الممثلة بشيفرة BCD .

٣ – تعليمات معالجة الخانات الثنائية.

٤ – تعليمات نقل التحكم : وهي التي تقوم بالتحكم بالتسلسل المنطقي لتنفيذ البرنامج .

٥ – تعليمات معالجة سلاسل الرموز – وهي تعليمات خاصة بمعالجة النصوص المختلفة .

٦ – تعليمات الاعتراض.

٧ – تعليمات التحكم بالمعالج .

وسوف نقوم بدراسة كل مجموعة من هذه المجموعات بالتفصيل .

اشارات العمليات

تعرف اشارات العمليات في لغة التجميع 8086 على انها رموز تستخدم في حقل المعامل في تعليمات المعالج او في التوجيهات ويمكن تقسيمها الى خمس مجموعات وهي :

١ – اشارات العمليات الحسابية

٢ – اشارات العمليات المنطقية

٣ – اشارات العلاقات

٤ – اشارات القيم الراجعة

٥ – ؟ ؟ عمليات التحديد

١ – اشارات العمليات الحسابية – وهي تلك الاشارات التي تفصل بين معاملين رقميين، وهذه الاشارات المستخدمة في لغة التجميع 8086 موضحة في الجدول وهذه بعض الامثلة على تلك الاشارات

X DW H+2

بهذا التعريف يخزن في موقع الذاكرة X العنوان الفعلي للرمز H زائد 2

Y DW Z -Z1

وبهذا التعريف يخزن في موقع Y ناتج طرح العنوان الفعلي لـ Z1 من Z ، لاعطاء البعد بين الموقعين .

Y1 EQU 30 x 20

الثابت Y1 يحتوي على ناتج ضرب 30x20

Y EQU 314/100

الثابتY يحتوي على ناتج القسمة 314/100 بدون باقي ، وبهذا تكون قيمة الثابت Y هي 3

R1 EQU 314 MOD 100

العملية MOD تعطي باقي قسمة 314/100 ، والقيمة هي 14 تخزن في الموقع R1 كثابت عددي .

العملية SHL تحريك محتويات معامل عددي لليسار بعدد معين من الخانات

M EQU M1 SHL 2

الثابت M يخزن فيه ناتج ازاحة الموقع M1 الى اليسار بمقدار خانتين .

M2 EQU M1 SHR 2

الثابت M2 يخزن فيه ناتج ازاحة الموقع M1 الى اليمين بمقدار خانتين

العمليات الحسابية

العملية	الشكل العام
+	value - 1 + value - 2 جمع value-2 , value-1
-	value - 1 - value - 2 طرح value - 2 من value - 1
*	value - 1 * value - 2
/	value - 1 / value - 2 تقسيم value 1 على value - 2 ، وناتج القسمة بدون باقي
MOD	value - 1 MOD value - 2 باقي القسمة
SHL	value SHL exp ازاحة value لليسار بمقدار exp
SHR	value SHR exp لليمين بمقدار exp

العمليات المنطقية

والعمليات المنطقية المستخدمة هي XOR , OR , AND التي تقوم على معاملين ، وهناك عملية النفي NOT التي تقوم على معامل واحد . والشكل العام لجميع هذه العمليات ممثل في الشكل .

عملية AND – وتنفذ هذه العملية على الخانات الثنائية المتناظرة في المعاملين الاول والثاني لاعطاء النتيجة .

مثال : 01000110 AND 01010100

V1 V2

نتيجة تنفيذ عملية AND على المعاملين هي 01000100

عملية - OR – وتنفذ هذه العملية على الخانات الثنائية المتناظرة في المعاملين الاول

والثاني لاعطاء النتيجة .

مثال 00100101 OR 11000011

نتيجة تنفيذ عملية OR على المعاملين هي 11100111

عملية XOR – وتنفذ عملية استثناء او هذه على الخانات الثنائية المتناظرة في المعاملين لاعطاء النتيجة

مثال 00110010 XOR 00111000

نتيجة تنفيذ هذه العملية ستكون 00001010

عملية NOT – تنفذ هذه العملية على معامل واحد ، ونتيجة لتنفيذها فان النتيجة ستكون المكمل لواحد لذلك المعامل

مثال NOT 11011000

النتيجة هي 00100111

امثلة على استخدام العمليات المنطقية في التعليمات

مثال

MOV AL ,00011010 AND 00011110

نتيجة لتنفيذ تعليمة MOV ، يتم نقل 00011010 الى المسجل AL

مثال MOV CL , NOT 00001101

يتم تخزين المكمل لواحد لـ 00001101 في CL .

عمليات العلاقات

تقارن هذه العمليات بين حقلين عددين أو بين عنوانين في نفس القطاع ، وتعطي هذه العمليات نتيجة لتنفيذها قيمة 0 اذا كانت العلاقة خاطئة (false) وقيمة 0FFFFH اذا كانت العلاقة صحيحة (TRUE)

مثال X EQU 30

اذا كانت X معرفة على انها ثابت يساوي 30

فان نتيجة تنفيذ التعليمة التالية هي تخزين 0 في AX

MOV AX , X LT 15

بما ان قيمة X ليست اقل من 15 فان العبارة المنطقية تكون خاطئة وفي هذه الحالة تكون قيمتها 0 ويتم تخزينها كما هو في المثال في المسجل AX

مثال

MOV AX , (0 AND 3) OR (0FFFF AND 7)

في هذا المثال يتم نقل 7 الى المسجل AX

عمليات القيمة الراجعة

تقوم هذه العمليات باعطاء معلومات عن المتغيرات والاسماء الاخرى المستخدمة في البرنامج .

عملية Offset

تقوم هذه العملية بحساب العنوان الفعلي للمتغير الذي يأتي بعدها

مثال Offset x – العنوان الفعلي لـ X

Offset x + 4 – العنوان الفعلي لـ x + 4

مثال العملية التالية تقوم بتحميل المسجل DX بالعنوان الفعلي لـ X

```
MOV  DX , Offset X
```

عملية SEG

وظيفة هذه العملية هي ايجاد عنوان القطاع الذي يقع فيه المتغير الذي يأتي بعد كلمة SEG

مثال SEG X – عنوان القطاع الذي يقع فيه المتغير X

```
MOV  DX , SEG X
```

تحمل المسجل DX بعنوان القطاع الذي يقع فيه المتغير X

عملية (LENGTH)

الوظيفة هي ايجاد عدد المدخلات المعرفة باستخدام المعامل DUP

مثال

```
X  DW  10 DUP (?)
MOV  DX , LENGTH X
```

التعليمة MOV تقوم بنقل 10 الى المسجل DX . والرقم 10 هو عدد المدخلات المعرفة بواسطة DUP.

عملية TYPE

الوظيفة هي ايجاد عدد البايتات المخصصة لتعريف موقع معين .

مثال

```
FLD    DB?
TAB    DW 20 Dup (?)
MOV   AX , TYPE FLD ; AX = 1
```

عدد البايتات المخصصة لتعريف المتغير FLD هو بايت واحد ولهذا يتم تحويل رقم ١ الى المسجل AX

اما التعليمة MOV AX , TYPE TAB

فتقوم بتحريك 2 الى المسجل AX ، لان TAB متغير معرف ككلمة تتكون من ٢ بايت.

عملية SIZE

وتحدد هذه العملية عدد البايتات المخصصة لمتغير معين ويتم ذلك بضرب (LENGTH)في TYPE

```
SIZE X
X   DW 10 DUP (?)
MOV AX , SIZE X ; AX = 20
```

العمليات المنطقية

	الشكل العام
AND	val - 1 AND val - 2
OR	val - 1 OR val - 2
XOR	val - 1 XOR val - 2
NOT	NOT val

عمليات العلاقات

العملية بلغة التجميع	التمثيل الرياضي	الشكل العام
EQ	=	operand - 1 EQ operand - 2
NE	#	operand - 1 NE operand - 2
LT	<	operand - 1 LT operand - 2
GT	>	operand - 1 GT operand - 2
LE	<	operand - 1 LE operand - 2
GE	>	operand - 1 GE operand - 2

عمليات القيمة الراجعة

العملية	
SEG	SEG Variable
OFFSET	OFFSET variable
TYPE	TYPE variable
SIZE	SIZE variable
LENGTH	LENGTH variable

عمليات اخرى (التحديد)

عملية PTR

وظيفة هذه العملية هي الحصول على محتويات موقع سعته Byte, Word,Dword أو عن طريق اعادة تعريف ذلك الموقع

مثال

```
X     DB   20H
      DB   30H
X1   DW 3045H
     MOV    AL, Byte PTR X1 ; AL = 45H
     MOV    CL, ByTE PTR X 1 + 1 ; CL = 30H
     MOV   AX, WORD PTR X ; AX =  2030H
```

عملية SHORT

وظيفة هذه العملية هي وصف عملية القفز القصير في جملة JMP عندما يكون مدى القفز يتراوح بين 128 - 127 +

JMP SHORT label

الوحدة الثانية

طرق العنونة

- فضاء عنوان الذاكرة
- فضاء عنوان موانىء الادخال والاخراج
- طرق العنونة المختلفة

طرق العنونة

تعرف طرق العنونة على انها الطريقة التي بواسطتها يتم الحصول على معاملات التعليمات ، والتي يمكن ان تكون مخزنة في أحد المواقع التالية

١ – في الذاكرة الرئيسية
٢ – مسجلات المعالج
٣ – موانىء الادخال والاخراج
٤ – تعطى فوريا في التعليمة.

باختلاف امكنة تخزين المعاملات تختلف طرق الوصول اليها ، ويمكن تقسيم تلك الطرق الى المجموعات التالية

١ – طرق عنونة الذاكرة للوصول للمعاملات المخزنة بها .
٢ – طرق عنونة بالمسجلات للوصول للمعاملات المخزنة بها .
٣ – طرق عنونة موانىء الادخال والاخراج .
٤ – طرق العنونة الفورية

يقوم المعالج بتحديد مكان تخزين المعامل باستخدام البايت الثاني من التعليمة وهو ثابت العنونة . وكما هو مبين في الشكل فان هذا البايت يحتوي على ثلاثة اجزاء وهي تحدد بمجملها اي مسجل سوف يستخدم كمعامل اول ، ومكان تخزين المعامل (الثاني في الذاكرة او في المسجل) .

والجدول يبين كيفية تحديد مكان التخزين بالاعتماد على محتويات الاجزاء المختلفة من البايت الثاني ، فإذا احتوت التعليمة على معامل واحد يتم تحديد مكان تخزينه بالاعتماد على الجزئين MOD و R/M ويكون الحقل MOD مساويا (١١) والحقل R/Mيشير الى رقم المسجل المستخدم كمعامل .

اذا كان الحقل MOD يحتوي على قيمة مخالفة لـ (١١) فان المعامل يمكن الحصول عليه باستخدام طرق العنونة التالية بالاعتماد على قيمة الحقل R/M : طرق العنونة المباشرة ، وغير المباشرة ، طرق العنونة النسبية ، والمفهرسة ، والمفهرسة النسبية .

اذا احتوت التعليمة على معاملين فان احدى هذه المعاملات سوف يكون مسجلا والمعامل الاخر يكون موقع ذاكرة او مسجل ، ولتحديد مكان تخزين المعامل الثاني يستخدم الحقل REG لتشير محتوياته الى رقم المسجل المستخدم كمعامل ثاني .

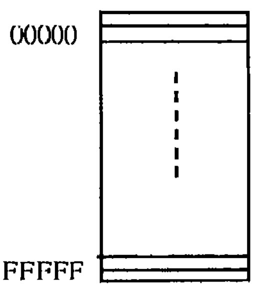
MOD	REG	R/M

بايت العنونه

MOD = 11			حساب العنوان الفعلي			
R/M	W = 0	W = 1	R/M	MOD = 00	MOD = 01	MOD = 10
000	AL	AX	000	(BX) + (SI)	(Bx) + (SI) + D8	(Bx) + (SI) + D16
001	CL	CX	001	(BX) + (DI)	(Bx) + (DI) + D8	(Bx) + (DI) + D16
010	DL	DX	010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	BL	BX	011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	AH	SP	100	(SI)	(SI) + D8	(SI) + D16
101	CH	BP	101	(DI)	(DI) + D 8	(DI) + D 16
100	DH	SI	100	Direct address	(BP) + D8	(BP) + D16
111	BH	DI	111	(BX)	(BX) + D 8	(BX) + D16

فضاء عنوان الذاكرة

يتكون ناقل العنوان في المعالج 8086 من ٢٠ خانة ثنائية ، وبواسطة هذا العنوان يمكن عنونة مليون موقع في الذاكرة ، والتي يمكن تصورها بسلسلة من المواقع المتتالية، سعة كل موقع هو بايت واحد . وتعنون المواقع من 0000 حتى FFFFF ، كما هو مبين في الشكل.

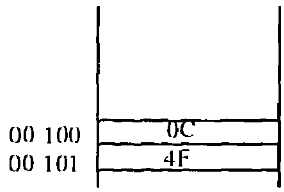


بواسطة هذا العنوان يستطيع المعالج الوصول الى التعليمات المخزنة في الذاكرة والمطلوب تنفيذها او الى معاملات تلك التعليمات . ويدعى هذا العنوان بالعنوان الفيزيائي .

تقسم الذاكرة الى مقاطع سعة كل مقطع هي 64KB ، يمكن للمبرمج ان يستخدم ٤ مقاطع في برنامجه في نفس اللحظة . يخزن عنوان بداية كل مقطع في احدى المسجلات التالية وهي SS , CS , DS , ES لاستخدام مقاطع اخرى من الذاكرة تغير محتويات DS و ES برمجيا .

يمكن للمعالج 8088 ان يقوم بمعالجة بيانات ذات طول بايت واحد او كلمة ، او كلمة مزدوجة . لمعالجة كلمة يحتاج المعالج للوصول الى موقعين متتالين في الذاكرة ،، بحيث ان

الموقع الذي عنوانه اقل سوف يكون البايت الاقل اهمية ، والموقع الذي عنوانه أكبر سوف يكون البايت الأكثر اهمية والشكل المجاور يوضح كيفية تخزين الكلمة 4 FOC في الذاكرة .



لحساب العنوان الفيزيائي للتعليمة المطلوب تنفيذها يستخدم مسجل قطاع التعليمات ، ومسجل مؤشر التعليمة IP ، ويكون العنوان الفيزيائي = (محتويات مسجل القطاع × ١٦) + (محتويات مسجل مؤشر التعليمة) اما العنوان الفيزيائي للمعامل المخزن في الذاكرة فيتم تحديده حسب المعادلة التالية

العنوان الفيزيائي للمعامل = العنوان الفعلي + (محتويات مسجل قطاع البيانات × ١٦) بعد ان تقوم وحدة EU بتمرير العنوان الفعلي الى وحدة BIU والتي بدورها تقوم بحساب العنوان الفيزيائي لذلك المعامل .

وبشكل عام نستطيع القول بأنه لتوليد عنوان موقع ذاكرة لا بد من تحديد

١ - عنوان بداية القطاع والموجود في احد مسجلات القطاع .

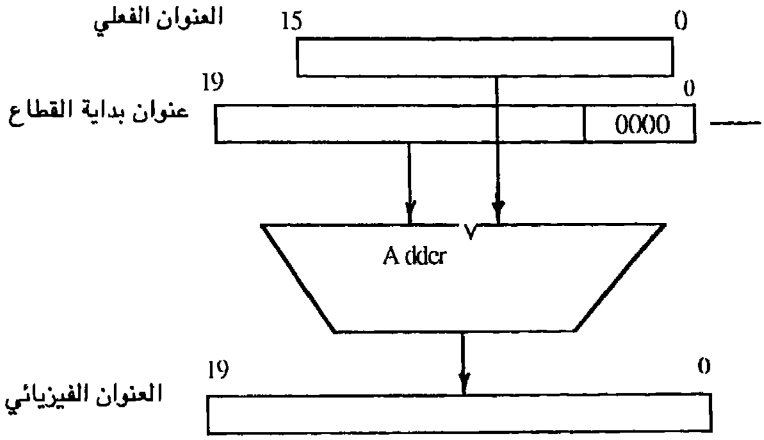
٢ - العنوان الفعلي للموقع او ما يسمى بعنوان الازاحة والذي يقصد به مقدار ازاحة ذلك الموقع عن بداية القطاع .

وعلى هذا الاساس يكون العنوان الفيزيائي

= (عنوان بداية القطاع × ١٦) + العنوان الفعلي

ويتم بازاحة محتوى مسجل القطاع الذي يحتوي ذلك العنوان اربع خانات لليسار .

والشكل التالي يوضح عملية توليد العنوان الفيزيائي في المعالج .



فضاء عنوان موانيء الادخال والاخراج

يمكن للمعالج 8086 ان يقوم بعنونة 2^{16} من موانيء الادخال والاخراج ، وعلى هذا الاساس يكون فضاء عنوان الموانيء 64 KB

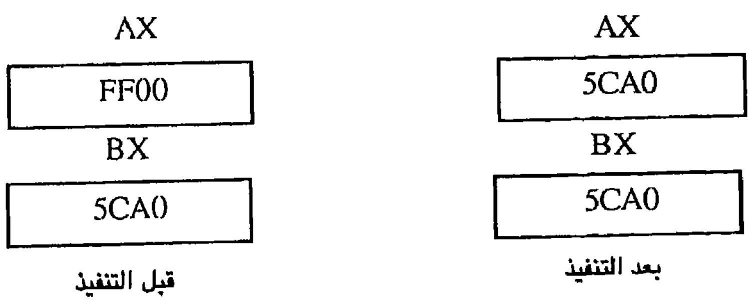
أما الان فسوف نقوم بدراسة طرق العنونة المختلفة بالتفصيل

١ – طريقة العنونة بالمسجلات : – باستخدام هذه الطريقة يقوم المعالج بالبحث عن المعامل في احد مسجلاته، وليس هناك حاجة للرجوع للذاكرة الرئيسية .

مثال

MOV AX , BX

تقوم هذه التعليمة بنقل محتويات المسجل BX الى المسجل AX ويبين الشكل حالة المسجلات المتأثرة بالعملية قبل وبعد تنفيذ التعليمة .



٢ - طرق العنونة الفورية .

المعامل المطلوب اجراء العمليات عليه يخزن في نفس التعليمة، وطول هذا المعامل يمكن ان يكون بايت واحد او كلمه .

مثال :

MOV BL , 20H

تنفيذ هذه التعليمة يؤدي الى نقل العدد 2OH الى المسجل BL اي انه لا حاجة للرجوع الى الذاكرة او الى المسجلات للحصول على البيانات ، وتنفيذ التعليمة يؤدي كذلك الى زيادة محتويات IP بمقدار 2 ، والشكل يبين حالة المسجلات المتأثرة قبل وبعد تنفيذ التعليمة .

	بعد التنفيذ	قبل التنفيذ
BL	20	50
IP	0002	0000

٣ - طرق عنونة الذاكرة

أ - طريقة العنونة المباشرة

تحتوي التعليمة على العنوان الفعلي للمعامل المطلوب اجراء العمليات عليه ، ويعطي العنوان في التعليمة على شكل اسم لموقع الذاكرة الذي يحتوي على المعامل ، ويقوم المترجم بتوليد عنوان ذلك الموقع بحساب مقدار ازاحة ذلك الموقع عن بداية القطاع المعرف فيه ، ويدعي هذا العنوان بعنوان الازاحة (OFFSET ADDRESS) .

مثال

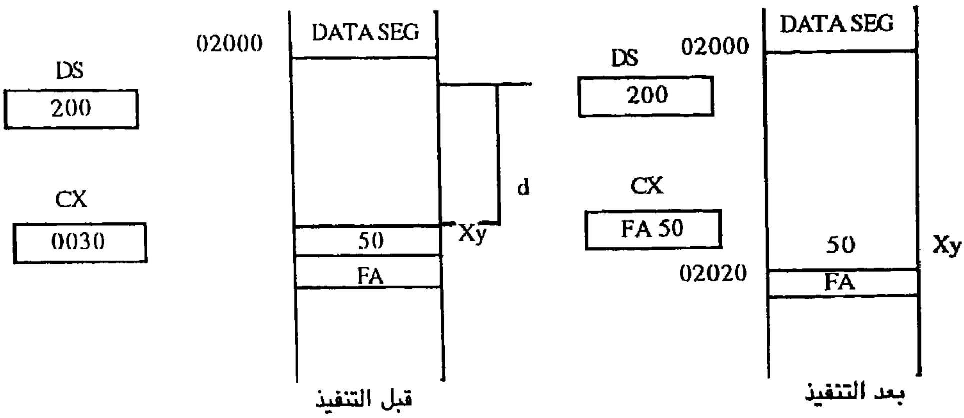
MOV CX,XY

تنفيذ هذه التعليمة يؤدي الى نقل محتوى الموقع XY الى المسجل CX فاذا كان عنوان الازاحة XY هو 20، وعنوان بداية القطاع الموجود فيه XY هو 200 مخزن في DS فإنه للحصول على العنوان الفيزيائي لـXY نعوض في المعادله التالية

phys - add = (محتوى مسجل القطاع × 16)+ Offsetaddress

phys - add = (200 x 16) + 20 = 2020

ويبين الشكل حالة الذاكرة والمسجل CX , DS قبل وبعد تنفيذ التعليمة



d - بعد الموقع Xy عن بداية قطاع البيانات

ب – طريقة العنونة المؤشرة

تستخدم في هذه الطريقة مسجلات التأشير SI و DI لحساب العنوان الفعلي للمعامل ويحتوي مسجل التأشير على مقدار الأزاحة عن بداية موقع معين في الذاكرة .

ولحساب العنوان الفعلي يتم جمع محتوى SI أو DI المستخدم في التعليمة مع عنوان الازاحة للموقع (Offset Address) .

العنوان الفعلي = (محتوى SI او DI) + عنوان ازاحة الموقع

مثال

MOV X [DI] , AL

تنفيذ هذه التعليمة يؤدي الى نقل محتوى AL الى موقع الذاكرة x والمؤشر عليه بـ DI

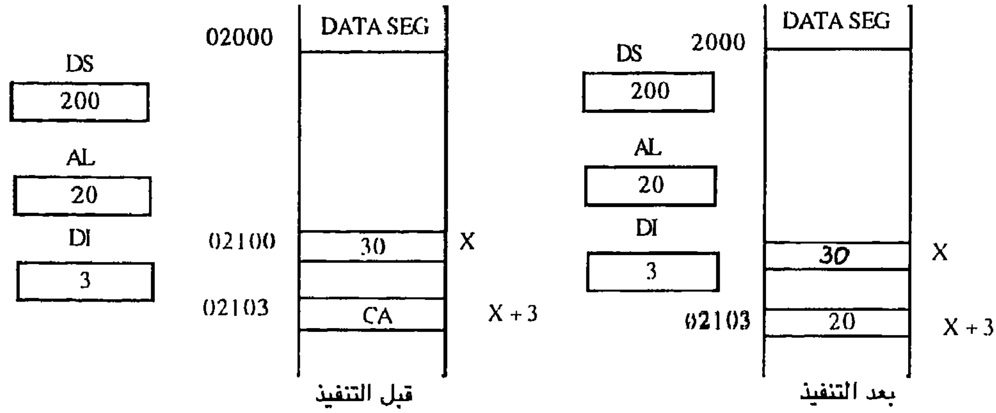
فاذا كان عنوان الازاحة لـ x هو 100، ومحتوى DS هو 200 ومحتوى DI هو 3

فيكون العنوان الفعلي للموقع المراد = 100 + 3 = 103

والعنوان الفيزيائي = (16 X 200) + العنوان الفعلي

= 103 + (16 x 200) = 2103

والشكل يبين تأثير تنفيذ هذه التعليمة على الذاكرة والمسجلات الاخرى .



جـ طريقة العنونة المفهرسة النسبية

للحصول على العنوان الفعلي للمعامل يتم جمع محتويات مسجل BX ومحتويات المسجل SI ومقدار الازاحة

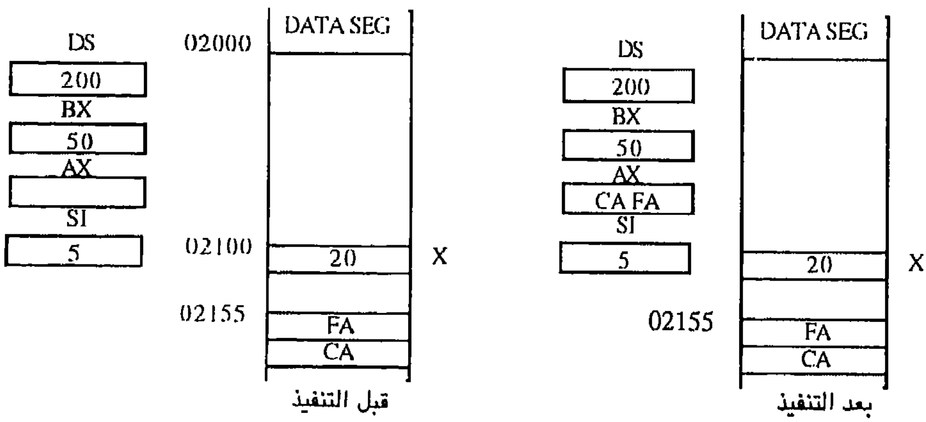
العنوان الفعلي = (BX) + (SI) + (مقدار الازاحة) .

اما العنوان الفيزيائي = العنوان الفعلي + (محتوى 16 X DS).

مثال MOV AX ,[BX][SI]+ X

تنفيذ هذه التعليمة يؤدي الى نقل محتويات الموقع الذي عنوانه الفعلي يساوي مجموع BX و SIومقدار الازاحة الى AX.

فاذا كان مقدار الازاحة للموقع X هو 100من بداية قطاع البيانات المعرف فيه ، ومحتويات BX تساوي 50 ومحتويات SI تساوي 5 فان تنفيذ هذه التعليمة يؤدي الى نقل محتويات الموقع الذي عنوانه الفعلي يساوي 155 الى AX . والشكل يبين تأثير تنفيذ التعليمة على الذاكرة وعلى المسجلات المختلفة .



د – العنونة النسبية

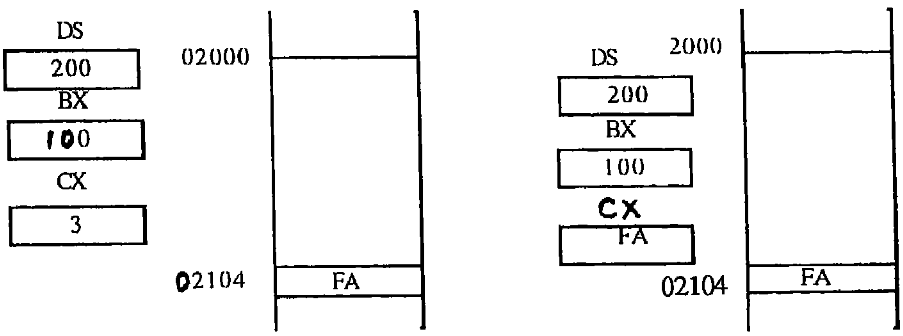
يتم الحصول على العنوان الفعلي للمعامل عن طريق جمع محتوى مسجل القاعدة BP أو BX ومقدار الازاحة الموجودة في التعليمة والتي تأتي عادة بعد اشارة +

مثال MOV CL ,[BX] + 4

حيث ان 4 في هذه التعليمة تشير الى مقدار الازاحة ولحساب العنوان الفيزيائي لذلك المعامل يستخدم المعالج المعادلة التالية

العنوان الفيزيائي = (محتوى 16 X DS) + (محتوى BX) + 4

ويبين الشكل تأثير تنفيذ هذه التعليمة على المسجلات والذاكرة



قبل التنفيذ بعد التنفيذ

هـ العنونة غير المباشرة

تحتوي التعليمة على العنوان الفعلي للمعامل مخزنا في احدى المسجلات التالية , DI SI , BP , BX

وللحصول على العنوان الفيزيائي للمعامل نعوض في المعادلة التالية:

Phys - add = (DS x 16) + (محتوى احد السجلات المذكورة)

فاذا كان المعامل مخزنا في الذاكرة في قطاع البيانات ، وعنوان الازاحة لهذا المعامل هو 50 ومخزن في المسجل BX

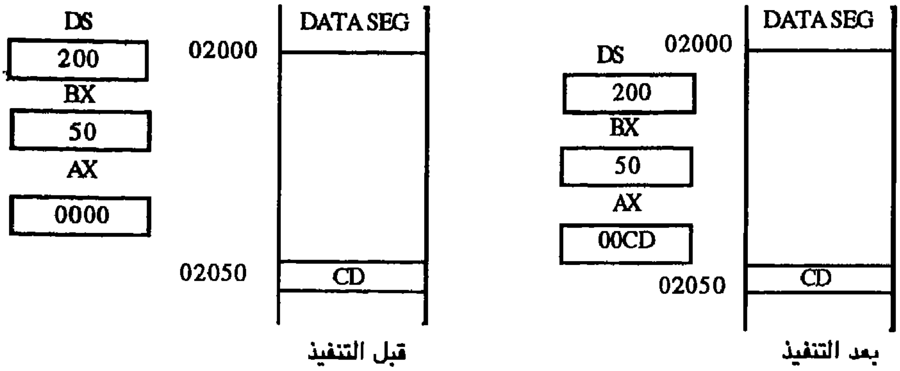
يكون العنوان الفيزيائي = (DS x 16) + (50)

فاذا كان محتوى DS = 200 فان العنوان الفيزيائي = 2050

مثال :

MOV AL, [BX]

تقوم هذه التعليمة بنقل محتويات الموقع المشار له بـ BX الى AL ، ويبين الشكل تأثير هذه التعليمة على الذاكرة والمسجل AL ، BX ، بعد التنفيذ



و - طريقة عنونة سلاسل الرموز

تستخدم تعليمات معالجة الرموز المسجل SI و DI لتخزين العنوان الفعلي لسلسلة الرموز فمثلا تنفيذ التعليمة MOVS يؤدي الى نقل سلسلة الرموز المعنونة بالمسجل SI الى مواقع الذاكرة المعنونة بـ DI

ز - طرق عنونة موانىء الادخال والاخراج

تستخدم طرق العنونة هذه مع تعليمات الادخا ل والاخراج (OUT, IN) حيث يحدد عنوان وحدة الادخال أو الاخراج في التعليمة نفسها او يشار اليه بواسطة المسجل DX .

مثال : I N AL , 15H

نقل محتويات الميناء والذي عنوانه 15الى المسجل AL

الوحـــدة الثـــالثـــة

تعريف البيانات

- تعريف البيانات
- تهيئة البرنامج للتنفيذ
- البرامج من نوع EXE و COM

تعريف البيانات

يستخدم مقطع البيانات لتعريف البيانات ، مناطق العمل ومناطق الادخال والاخراج وتتنوع البيانات المستخدمة في لغة التجميع ومن اهم انواع البيانات المستخدمة نورد ما يلي :

١ - الثوابت الرقمية (Numeric CONSTANTS)

حيث تستخدم هذه البيانات في اجراء كافة العمليات الحسابية وفي عنونة الذاكرة وعند معالجة هذه البيانات تتم عملية تحويلها الى نظام السادس عشر ويحتفظ بها بشكل معكوس (Reverse) بدأ من بايت معين ثم التالي ومن انواع هذه الثوابت ما يلي:

- العشرية (Decimal)

وهي الثوابت التي تحتوي على الارقام العربية (9 - 0) ويتم استخدام هذه الثوابت باستخدام الحرف D (أو بدونه) في نهاية الرقم فمثلا الثابت 123D أو 123 ثابتا عشريا وتتم عملية تحويل هذه الثوابت الى نظام السادس عشر عند عملية المعالجة .

- السادس عشرية (Hexadecimal)

تستخدم في هذه الثوابت الارقام (O...F) متبوعة بالحرف H ولتمييز الاسماء عن الثوابت السادس عشرية لا بد ان يبدأ هذا الثابت بالرقم [0] فمثلا الثوابت , OF5DH 5EH هي ثوابت سادس عشرية ويتم تخزينها في الذاكرة بشكل معكوس فالرقم الاول يخزن في بايت وبصورة 5E اما الرقم الثاني فيخزن في ٢ بايت وبصورة 5DOF على اعتبار ان OF تخزن في البايت المخصص الاول و 5D في البايت التالي .

- الثنائية (Binary)

حيث تستخدم الارقام الثنائية (0,1) في هذه الثوابت على ان تكون متبوعة بالحرف B وعادة ما تستخدم هذه الثوابت في عمليات الفحص والعمليات المنطقية.

- الثماينة (Octal)

وهي الثوابت التي تستخدم الارقام الثمانية (0..7) على ان تكون متبوعة بالحرف Q

٢ - السلاسل الابجدية (Character string)

تستخدم السلاسل الابجدية لوصف بيانات لا تستخدم في العمليات الحسابية كأسماء الاشخاص مثلا وقد تحتوي السلاسل على اي نوع من الرموز المستخدمة في لغة التجميع (كالاحرف والارقام والرموز والاشارات) على ان تكون السلسلة موضوعة بين علامات الاقتباس (" " أو ' ').

يتم تخزين البيانات الابجدية باستخدام الشيفرة الامريكية (ASCII Code) ويمكن التعامل مع البيانات وذلك من خلال تعريفها في مقطع البيانات كما ويمكن التعامل مباشرة مع الثوابت (بدون تعريفها) مباشرة وذلك اثناء تنفيذ هذه التعليمات . وعند تعريف البيانات في مقطع البيانات يتم الرجوع اليها في البرنامج وذلك باستخدام اسمها حيث تنظم البيانات في حقول على ان يعطى كل حقل اسما كما يلي .

fieldname Dn expression

التعليمات المخصصة لتعريف البيانات

تمتلك لغة التجميع بعض التعليمات الخاصة بوصف البيانات ومن هذه التعليمات نورد :

١ – تعريف البايت (Define Byte DB)

تستخدم التعليمة DB لتعريف كافة انواع البيانات المستخدمة في البرنامج حيث يتم تخصيص بايت واحد لكل عنصر من عناصر البيانات .

– تستخدم DB لتعريف السلاسل الابجدية (يتم تعريف السلاسل فقط بهذه التعليمة) ويمكن للسلسلة المعرفة بهذه التعليمة ان تأخذ اطوالا متنوعة واقصى طول للسلسلة المعرفة بهذه التعليمةمحدد بطول السطر وعند استخدام هذه التعليمة لتعريف الابجديات يراعى وضع الابجدية بين علامات الاقتباس .

– البيانات العشرية المعرفة بهذه التعليمة محصورة بين الرقم 127+ و 128_ (على اعتبار ان هناك بايت واحد مخصص لعنصر البيانات) .

– البيانات السادس عشرية محصورة بين الرقم الموجب 7F والرقم السالب FF على اعتبار ان الارقام السالبة محصورة بين الرقم 80 والرقم FF

٢ – تعريف الكلمة (Define word DW)

تعرف عناصر البيانات والتي تحتاج الى ٢ بايت بهذه التعليمة حيث يتم تخصيص كلمة كاملة (٢ بايت) لعنصر البيانات (لا تستخدم هذه التعليمة لتعريف الابجديات) وعند استخدام هذه التعليمة يجب مراعاة ما يلي:

– اكبر رقم عشري موجب هو 32767 وسالب هو 32768 (على اعتبار انه يتم تخصيص ١٦ خلية ثنائية لعنصر البيانات) .

– اكبر رقم سادس عشري موجب هو7 FFF (الارقام الموجبة محصورة بين- 0000) 7FFF) واكبر رقم سالب هو FFFF (الارقام السالبة محصورة بين (8000 - FFFF)

يتم تخزين البيانات بالنظام السادس عشري بشكل معكوس.

٣ – تعريف الكلمة المضاعفة (Define Doubleword DD)

يتم تخصيص ٤ بايت لعنصر البيانات ويمكن استخدام هذه التعليمة لوصف الابجديات على ان لا يزيد طولها عن حرفين .

٤ – تعريف رباعية الكلمة (Define Quadword DQ)

تخصص بهذه التعليمة ٤ كلمات (٨ بايت) لعنصر البيانات .

٥ – تعريف العشرة بايت (Define ten byte DT)

يتم تخصيص ١٠ بايت لعنصر البيانات وكأمثلة على هذه التعليمات وكيفية استخدامها نورد البرنامج التالي.

```
         PAGE 30,80
TITLE  CHAP2  DATA DEFINITION
DATASG  SEGMENT PARA 'DATA'
;=====   DEFINE DATA FIELDS
;        ------------------
;==============================================
;        DEFINE BYTE - DB
;        ---------------
FLD1           DB      'ASSEMBLER'   ;CHARACTER STRING
FLD2           DB      55            ;DECIMAL CONSTANT
FLD3           DB      10H           ;HEX CONSTANT
FLD4           DB      11011B        ;BINARY CONSTANT
FLD5           DB      01,'SON',02,'MON'   ;TABLE
FLD6           DB      '12345'       ;NUMBERS ARE CHAR.
FLD7           DB      10 DUP(0)     ;TEN ZEROS
FLD8           DB      ?             ;UNINITIALIZED
;==============================================
;           DEFINE WORD-DW
;           ---------------
FLD9           DW      0FFF0H        ;HEX CONSTANT
FLD10   DW     000111B       ;BINARY CONST.
FLD11   DW     FLD6          ;ADDRESS CONST.
FLD12   DW     3,4,5         ;THREE CONST.
FLD13   DW     5 DUP('*')    ;FIVE STARS
;==============================================
;           DEFINE DOUBLEWORD - DD
;           ---------------------
FLD14   DD     ?             ;UNINITIALIZED
FLD15   DD     'AB'          ;CHARACTER STRING
FLD16   DD     6543          ;DECIMAL VALUE
FLD17   DD     FLD3 - FLDFLD4 ;DIFF BETW ADDRESSES
FLD18   DD     16,17         ;TWO CONSTANT
;==============================================
;           DEFINE QUADWORD - DQ
;           ---------------------
FLD19   DQ     ?             ;UNINITIALIZED
FLD20   DQ     05D23         ;HEX CONSTANT
FLD21   DQ     2314          ;DECIMAL CONSTANT
;==============================================
;           DEFINE TEN BYTE - DT
;           ---------------------
FLD22   DT     ?             ;UNINITIALIZED
FLD23   DT     'ZIAD'        ;CHAR STRING
;==============================================
;       END DATA DEFINITION
;       ------------------
DATASG  ENDS
               END
```

وكما اسلفنا سابقا فان التعليمات المنفذة يمكن ان تتعامل مباشرة مع المسجلات او مواقع الذاكرة فوريا ولكن يراعى عند هذا ان لا تزيد قيمة البيانات عن طول المسجل المستخدم كما ويمكن اعطاء حقول البيانات المعرفة قيما داخل البرنامج وذلك من خلال استخدام امر التخصيص التالي

fieldname EQU value

فمثلا يمكن ان يخصص الرقم ١٥ للحقل FLD2 ومن ثم يمكن الرجوع الى القيمة (١٥) باستخدام هذا الحقل

```
FLD2 EQU 15
.
.
.
MOV CX , FLD2
```

تهيئة البرنامج للتنفيذ

عند كتابة البرامج التنفيذية (EXE Type) لا بد من مراعاة بعض الامور الهامة والمتعلقة بتهيئة البرنامج للتنفيذ ومن هذه الامور ما يلي :

١ – استخدام الامر ASSUME حيث يقوم هذا الامر باخبار المترجم عن المقاطع المستخدمة والمسجلات المرتبطة بهذه المقاطع وذلك ليتم حساب العنوان الفعلي للتعليمات والبيانات المستخدمة في البرنامج .

٢ – يسبق البرنامج التنفيذي عادة مقطع خاص (مقدمة البرنامج) PROGRAM SEGMENT PREFIX (PSP) بطول ٢٥٦ بايت (100H) يستخدم للاحتفاظ بمعلومات خاصة عن البرنامج كحجم الذاكرة المتوفرة ، عنوان نقطة النهاية والخروج من البرنامج وغيرها من معلومات هامة ويقوم برنامج التحميل (Loader)باستخدام المسجل DSR لبناء هذا المقطع ويجب الاحتفاظ بعنوان هذا المقطع عن طريق دفع محتويات هذا المسجل في مسجل الحزمة (PUSH DS)

٣ – تحديد البداية الفعلية (العنوان صفر) للمقطع SS و CS وذلك بدفع هذا العنوان في الحزمة .

٤ – تحديد عنوان البيانات وذلك بتخزين هذا العنوان في المسجل DSR

٥ – العودة الى نقطة البداية وذلك باسترجاع العنوان من SS (الامر RET) . والتعليمات التالية توضح هذه المفاهيم

```
; EXE-PROGRAM INITIALIZATION
;==========================
CODESG  SEGMENT PARA 'CODE'
BEGIN
;   1)
                    ASSUME CS:CODESG,DS:DATASG,SS:STACKSG
;   2)
                    PUSH DS            ;STORE DS IN STACK
;   3)
                    SUB  AX,AX
                    PUSH AX            ;STORE ZERO IN STACK
;   4)
                    MOV AX,DATASG      ;STORE ADDRESS OF
                    MOV DS,AX          ;DATASG IN DS
;                   .
;                   .
;                   .
;   5)
                    RET                ;RETURN TO DOS
BEGIN   ENDP
CODESG  ENDS
                    END     BEGIN
```

وهناك نوع آخر من البرامج (COM Type) وتختلف هذه البرامج عن البرامج من نوع (EXE) في الامور التالية :

١ – حجم البرنامج COM مقيد بـ ٦٤ كيلو بايت بينما حجم البرنامج EXE قد يزيد عن هذا الرقم .

٢ – يتطلب البرنامج من نوع EXE مقدمة (program header) بحجم ٥١٢ بايت تستخدم لاعطاء معلومات عامة عن البرنامج كحجمه والعنوان الفعلي ، عدد التعليمات ، وعناصر البيانات وغيرها في حين لا حاجة لهذه المقدمة للبرنامج من نوع COM.

٣ – لا بد من تعريف مقطع الحزمة (SS) في البرنامج (EXE) في حين انه لا داعي لاجراء هذا التعريف في البرنامج COM حيث يقوم هذا البرنامج اوتوماتيكيا بتوليد هذه الحزمة .

٤ – تعرف البيانات (ان لزم الامر) في البرنامج من نوع COM في مقطع التعليمات (CS)

٥ – تهيئة هذا البرنامج تتم باستخدام الامر ASSUME متبوعة بالعنوان (100H) (حجم PSP) ويتم تحقيق هذا باستخدام الامر

ORG 100H

والبرامج التالية توضح اهم الفروقات بين البرنامج من نوع EXE و COM

```
                PAGE 30,50
TITLE   EXE-PROGRAM
;=============================================
STACKSG SEGMENT PARA STACK 'STACK'
                DW 32 DUP(?)
STACKSG ENDS
;==============================================
DATASG  SEGMENT PARA 'DATA'
FL1             DW 200
FL2             DW 100
FL3             DW ?
DATASG  ENDS
;==============================================
CODESG  SEGMENT PARA 'CODE'
BEGIN   PROC    FAR
                ASSUME CS:CODESG,DS:DATASG,SS:STACKSG,ES:NOTHING
                PUSH DS
                SUB AX,AX
                PUSH AX
                MOV AX,DATASG
                MOV DS,AX
;           ====================== INITIALIZATION END
                MOV AX,FL1
                ADD AX,FL2
                MOV FL3,AX
                RET
BEGIN   ENDP
CODESG  ENDS
                END     BEGIN
```

```
                PAGE 30,60
TITLE   COM-PROGRAM
CODESG  SEGMENT PARA 'CODE'
                ASSUME CS:CODESG,DS:CODESG,SS:STACKSG,ES:CODESG
                ORG 100H        ;START AT END OF PSP(256 BYTE)
BEGIN:  JMP     MAIN
;..... DATA TO BE DEFINED IN CODESG................
FL1             DW 200
FL2             DW 100
FL3             DW ?
;==============================================
MAIN            PROC    NEAR
                MOV AX,FL1
                ADD  AX,FL2
                MOV FL3,AX
                RET
MAIN            ENDP
CODESG  ENDS
                END     BEGIN

B>
```

الوحدة الرابعة

تعليمات نقل البيانات

- تعليمات التحريك العامة
- تعليمات الادخال والاخراج
- تعليمات نقل العنوان

تعليمات نقل البيانات

وظيفة هذه المجموعة من التعليمات هي عملية تحريك البيانات بين مواقع الذاكرة والمسجلات في المعالج وموانيء الادخال والاخراج.

تعليمات التحريك العامة

سنقوم بتقسيم هذه المجموعة الى 5 اصناف وهي

أ - تعليمات تحريك البيانات عامة الاغراض وهي موضحة في الجدول المبين في الشكل

	التعليمة	الشكل العام
١	MOV	MOV des , source
٢	PUSH	PUSH source
٣	POP	POP des
٤	XCHG	XCH G des , source
٥	XLAT	XLAT table - name

تعمل تعليمة MOV على نقل البيانات من الحقل المرسل الى الحقل المستقبل ويمكن لهذه التعليمة ان تأخذ الصيغ العامة التالية

١) MOV R, R1

٢) MOV M,R

٣) MOV R, M

٤) MOV M, im-d

٥) MOV R, im-d

في هذه الصيغ R1, R - تشير الى احد مسجلات المعالج

M موقع ذاكرة

im-d بيانات رقمية .

هنالك بعض الاستثناءات في استخدام تعليمةMOV وهي

١ – لا يجوز نقل محتوى موقع ذاكرة الى موقع ذاكرة اخر مباشرة في نفس التعليمة.

٢ – لا يمكن نقل البيانات فوريا الى اي من مسجلات القطاعات باستخدام هذه التعليمة .

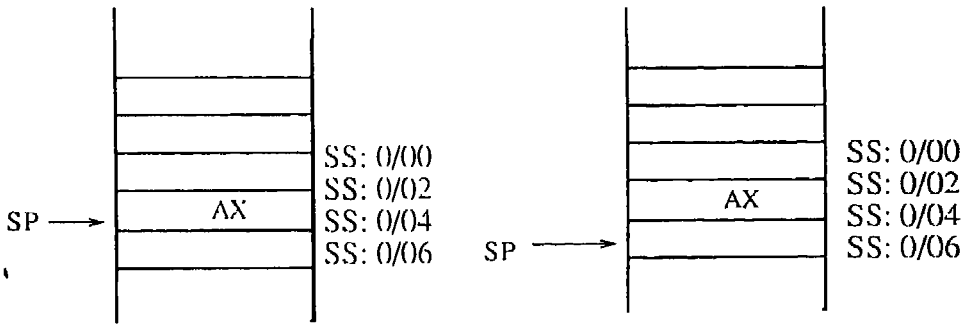
٣ – لا يمكن نقل محتوى مسجل قطاع الى مسجل قطاع اخر .

٤ – لا يمكن استخدام المسجل ES كحقل مستقبل في هذه التعليمة .

مثال :

POP AX

عند تنفيذ هذه التعليمة يقوم المعالج بقراءة الكلمة المشار لها بـ SP ونقلها الى المسجل AX ، ومن ثم زيادة محتوى SP بمقدار 2 والشكل التالي يوضح حالة الكومة قبل وبعد تنفيذ هذه التعليمة



ب – تعليمة XCHG

وظيفة هذه التعليمة هي القيام بتبديل محتويات حقلين والشكل العام

XCHG des , sour

حيث يمكن تبديل محتوى مسجل مع محتويات موقع ذاكرة

مثل

XCHG AX , BX

ويستثنى من هذه المسجلات مسجلات القطاعات وهي CS, DS , SS , ES ولا تجوز عملية تبديل محتويات موقعين في الذاكرة .

جـ – XLAT

وظيفة هذه التعليمة هي البحث عن عنصر في مصفوفة باستخدام مؤشر ذلك العنصر . وتحميل ذلك العنصر في المسجل AL

قبل البدء بتنفيذ هذه التعليمة يحمل عنوان بداية المصفوفة في BX وقيمة المؤشر في AL

مثال : للبحث في المصفوفة A عن قيمة العنصر العاشر

نقوم بالعمليات التالية

```
MOV AL, 10
MOV BX , Offset A
XLAT
```

ومن الامثلة على الحالات المذكورة غير المسموح بها

MOV X , S – عملية نقل غير مسموح بها

MOV DS , SH – عملية نقل غير مسموح بها

MOV ES , DS – عملية غير مسموح بها

MOV ES . AX – عملية غير مسموح بها

د – تعليمة PUSH

تقوم هذه التعليمة بنقل محتوى الحقل المرسل والذي يتكون من 16 خانة ثنائية في قمة الكومة .

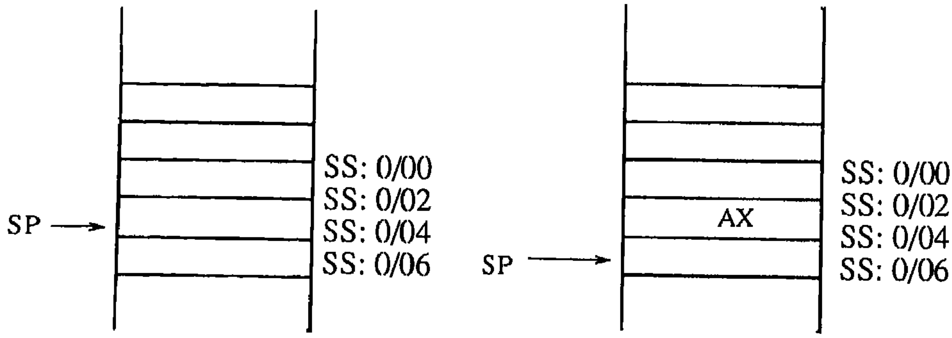
الشكل العام لهذه التعليمة هو

PUSH Source

حيث ان Source – تشير الى الحقل المرسل والذي يمكن ان يكون مسجلا من مسجلات المعالج بطول 16 بت ، او موقع ذاكرة بطول 16بت وعند تنفيذ هذه التعليمة يتم طرح 2 من مؤشر الكومة SP ومن ثم تخزين محتوى الحقل المرسل في موقع الذاكرة المشار له بمؤشر الكومة SP

مثال :

PUSH AX



والشكل في الاعلى يوضح حالة الكومة قبل وبعد تنفيذ التعليمة

هـ - تعليمة POP

تقوم هذه التعليمة بعمل عكسي لعمل PUSH ، حيث تعمل على استرجاع الكلمة المشار لها بمؤشر الكومة SP ، وتخزينها في الحقل المستقبل والذي يمكن ان يكون مسجل او موقع ذاكرة

الشكل العام

POP dest

حيث ان dest تشير الى الحقل المستقبل اي الحقل الذي يتم فيه تخزين الكلمة التي تم استرجاعها من الكومة .

تعليمة PUSH F

تعمل هذه التعليمة على تخزين محتويات مسجل الرايات في ذاكرة الكومة

تعليمة POP F

تعمل هذه التعليمة على استرجاع الكلمة المشار لها بـ SP في مسجل الرايات .

تعليمات الادخال والاخراج

الشكل العام لتعليمة الادخال هو

IN accumulator , port

حيث ان accumulator – هو AL لنقل بايت ، و AX لنقل كلمة من ميناء الادخال

port – وهو عنوان ميناء الادخال ، ويمكن ان يكون رقماً قيمته تقع بين 0 , 255 . ويمكن ان يكون المسجل DX

مثال

IN AL , 200

ادخال بايت واحد من البيانات الى المسجل AL من ميناء الادخال رقم 200

IN AL , DX

ادخال بايت واحد من البيانات الى المسجل AL من ميناء الادخال والذي عنوانه محتوى DX

اما الشكل العام لتعليمة الاخراج فهو

OUT Port, accumulator

مثال

OUT 20H , AL

هذه التعليمة تقوم باخراج محتويات AL الى ميناء الاخراج والذي عنوانه 20H

مثال

OUT DX, AX

اخراج محتويات AX الى ميناء الاخراج والذي عنوانه محتوى DX.

تعليمات نقل العنوان

تعمل هذه المجموعة على نقل عنوان المتغيرات المعرفة في الذاكرة.

تعليمة LEA– تقوم هذه التعليمة بنقل عنوان الازاحة للمعامل الى احدى مسجلات الاغراض العامة AX , BX , CX , DX او الى SP , BP , SI , DI

الشكل العام لهذه التعليمة

LEA reg 16 , mem 16

reg 16 - وهو احدى المسجلات المذكورة في الاعلى.

mem 16 – اسم موقع في الذاكرة.

مثال

LEA SI , X

تعمل هذه التعليمة على نقل عنوان الموقع x الى المسجل si

مثال :

LEA AX , X [SI]

تقوم بنقل عنوان الازاحة لـ X + DI في AX

فاذا كان محتوى SI هو 5 فانه يتم نقل العنوان 5 الى AX

تعليمة LDS

تقوم هذه التعليمة بالبحث عن كلمة مزدوجة معرفة في الذاكرة ، وتحميل الـ ١٦ خانة الاكبر اهمية في مسجل معين ، والـ ١٦ خانة الاقل اهمية في DS

الشكل العام هو

LDS reg 16 , mem 32

mem 32 – موقع ذاكرة معرف لكلمة مزدوجة .

reg 16 – مسجل يتسع لـ ١٦ خانة ثنائية .

مثال

LDS BX , X

تعمل هذه التعليمة على نقل الخانات الاقل اهمية من الموقع x الى DS والخانات الاكبر اهمية الى . BX.

تعليمة LES

تعمل هذه التعليمة مثل تعليمة LDS ، والاختلاف انه يتم تحميل ES بدل DS .

٤ – تعليمات نقل محتويات مسجل الرايات

تعليمة LAHF – تعمل هذه التعليمة على نقل محتويات مسجل الرايات , SF , ZF AF , PF , CF في المسجل AHفي الخانات (7, 6, 4, 2, 0)

تعليمة SAHF – تعمل على نقل محتويات AH في الخانات (7, 6, 4, 2, 0)الى مسجل الرايات .

مثال : – اكتب برنامج يقوم بحجز خمس مواقع في الذاكرة ، وتصفير تلك المواقع .

برنامج رقم (١)

```
STACK   SEGMENT STACK PARA 'STACK'
                DW      32 DUP(?)
STACK   ENDS
DAT             SEGMENT
X               DB 5 DUP (?)
DAT             ENDS
COD             SEGMENT
                ASSUME CS:COD,DS:DAT,SS:STACK,ES:NOTHING
                PUSH DS
                SUB AX,AX
                PUSH AX
                MOV AX,DAT
                MOV DS,AX
                MOV BX,OFFSET X
LL:           MOV X[BX],0
                INC BX
                CMP BX,OFFSET X +5
                JNZ LL
                HLT
COD             ENDS
                END
```

مثال : – اكتب برنامج يقوم بتبديل العناصر المتناظرة في المصفوفة X والمصفوفة Y

برنامج رقم (٢)

```
STACK    SEGMENT STACK PARA 'STACK'
                 DW 64 DUP (?)
STACK    ENDS
DAT              SEGMENT PARA 'DATA'
X                DB 8,9,50,30,20,70
Y                DB 6,80,44,77,32,55
DAT              ENDS
COD              SEGMENT PARA 'CODE'
                 ASSUME CS:COD,DS:DAT,SS:STACK
                 PUSH DS
                 SUB AX,AX
                 PUSH AX
                 MOV AX,DAT
                 MOV DS,AX
                 MOV SI,OFFSET X
                 MOV DI,OFFSET Y
LL:          MOV DL,X[SI]
                 XCHG [DI],DL
                 MOV X[SI],DL
                 INC SI
                 INC DI
                 CMP SI,OFFSET X +6
                 JB LL
                 HLT
COD              ENDS
                 END
```

الوحـدة الخـامسة

برمجة العمليات الحسابية

- جمع وطرح الارقام الثنائية
- جمع الارقام متعددة البايت
- جمع وطرح الارقام العشرية
- ضرب الارقام الثنائية
- قسمة الارقام الثنائية
- ضرب وقسمة الارقام الممثلة بنظام ASCII
- تحويل الارقام من نظام ASCII الى BCD
- تحويل الإرقام من نظام ASCII الى الثنائي
- تحويل الارقام من النظام الثنائي الى ASCII

برمجة العمليات الحسابية

تحتوي لغة التجميع عادة على مجموعة من التعليمات المخصصة لتنفيذ العمليات الحسابية الاساسية : الجمع ، الطرح ، الضرب والقسمة .

يقوم الحاسب بتنفيذ هذه العمليات على الارقام الثنائية التي يحدد طولها ببايت واحد او كلمة واحدة . لذا تتطلب معالجة الارقام التي يزيد طولها عن هذه القيم معاملة خاصة ، خصصت هذه الوحدة لتوضيح طرق برمجة العمليات الحسابية المختلفة باستخدام التعليمات الحسابية المتوفرة في لغة التجميع .

يستطيع المعالج الدقيق تنفيذ العمليات الحسابية على الارقام الثنائية المؤشرة وغير المؤشرة وكذلك على الارقام العشرية غير المؤشرة.

يحدد طول الارقام الثنائية بـ 8 او 16 ثنائية . ففي حالة الارقام الثنائية غير المؤشرة يحتل الرقم جميع الخانات ويتراوح بين 0÷ 255 (اذا كان طول الرقم 8 خانات) أو0 ÷65536 (اذا كان الرقم طوله 16 خانة) . تحتل الاشارة اقصى خانة من اليسار في الارقام المؤشرة (الخانة رقم 7 والخانة رقم 15) وبهذا تتراوح قيمة الارقام من 127 الى 128 - (8 خانات) او من 32706 الى 32768 - (16 خانة).

تمثل الارقام العشرية العشرية في ذاكرة الحاسب باحدى الصيغتين : -

- الصيغة المحزومة)Packed Form) ، حيث يحتوي البايت على رقمين BCD يحتل كل رقم الى 4 خانات ثنائية.

- الصيغة المفكوكة (Unpacked) حيث يحتوي البايت على رقم BCD واحد ، يحتل هذا الرقم النصف الايمن من البايت . تكون الخانات الاربع اليسرى اما صفرا في حالة الضرب والقسمة او اية قيمة اخرى في حالات الجمع والطرح . نمثل الرقم 1990 في الصيغتين : -

1	9

0	0	0	1	1	0	0	1

9	0

1	0	0	1	0	0	0	0

تمثيل الرقم 1990 في الصيغة المحزومة

البايت الرابع	البايت الثالث	البايت الثاني	البايت الاول
1	9	9	0
0001	1001	1001	0000

تمثيل الرقم 1990 في الصيغة المفكوكة

واخيرا تجدر الاشارة الى ان المعالج الدقيق 8088/8086 يقوم بتنفيذ العمليات الحسابية الاساسية على افتراض ان الارقام ممثلة في النظام الثنائي . يؤدي هذا الافتراض الى الحصول على نتائج غير صحيحة اذا كانت الارقام ممثلة في نظام غير النظام الثنائي . تتطلب عملية الحصول على النتائج الصحيحة تنفيذ تعليمات التعديل Adjust instructions على النتائج الخاطئة.

– جمع وطرح الارقام الثنائية

ينفذ المعالج الدقيق عمليات جمع وطرح الارقام الثنائية باستخدام التعليمات ADD و SUB . يمكن استخدام هذه التعليمات لمعالجة (جمع او طرح) رقمين فقط في نفس الوقت بحيث يكون طول كل منهما بايت واحد او كلمة واحدة . الشكل العام لهذه التعليمات هو

ADD المعامل الاول والمعامل الثاني

SUB المعامل الاول والمعامل الثاني

يتم ، حسب تعليمة ADD جمع المعاملين وتخزين المجموع في المعامل الثاني . يتم حسب تعليمة SUB طرح المعامل الاول من المعامل الثاني وتخزين ناتج الطرح في المعامل الثاني ، يخزن المعامل الاول في : –

– احد المسجلات عامة الاغراض General - purpose register
– احد مواقع الذاكرة Memory Location
– نفس التعليمة

اما المعامل الثاني فيمكن تخزينه في احد المسجلات عامة الاغراض او في احد مواقع الذاكرة . غير انه يشترط عدم كتابة تعليمة ADD او SUB مع معاملين مخزنين في الذاكرة : ADD BYTE2 لجمع او طرح رقمين مخزنين في الذاكرة ينقل احد الرقمين الى احد المسجلات ثم تكتب التعليمة بمشاركة هذا المسجل :

```
MOV   AL , BYTE1
```

MOV AL , BYTE2

يشترط ايضا استخدام تعليمة ADD او SUB مع معاملات متساوية في الطول ، اي يمكن جمع بايت مع بايت او كلمة مع كلمة ولا يجوز جمع بايت مع كلمة . المثال التالي يحتوي على اشكال مختلفة للتعليمات ADD و SUB

```
;    ********    ADDITION  &  SUBTRACTION   INSTRUCTIONS
;    ********                EXAMPLES
                 PAGE  60,60
CODE             SEGMENT  PARA ' CODE '
                 ASSUME   CS:CODE, DS:CODE, SS:CODE
                 ORG      100H
BEGIN:  JMP        SHORT MAIN
;  ******  Data    definitions        ****************
BYTE1   DB         60H
BYTE2   DB         30H
BYTE3   DB         00H
WORD1   DW         6000H
WORD2   DW         2000H
WORD3   DW         0010H
;**************************************************
MAIN             PROC
                 MOV       AL, BYTE1
                 MOV       BL, BYTE2
                 ADD       AL, BL       ;reg + reg
                 ADD       AL, BYTE3    ;reg + memory
                 ADD       BYTE1,BL     ;memory + reg
                 ADD       BL, 05H      ;immediate + reg
                 ADD       BYTE1, 12H ;immediate + mem
                 MOV       AX, WORD1
                 MOV       BX, WORD2
                 SUB       AX, BX       ;reg - reg
                 SUB      AX, WORD3     ;reg - mem
                 SUB       WORD1, BX    ;mem - reg
                 SUB       BX, 2560H    ;reg - immed
                 SUB       WORD1, 357H;mem - immed
                 RET
MAIN    ENDP
CODE    ENDS
                 END       BEGIN
```

قد تؤدي عملية جمع بايت مع بايت في كثير من الاحيان الى الحصول على مجموع لا يمكن تخزينه في بايت واحد . تسمى الحالة بالفيض . لنفرض ان المسجل AL يحتوي الرقم 60H . ان تنفيذ التعليمة

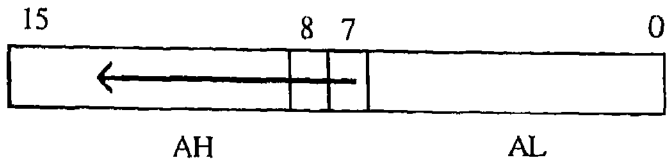
ADD AL , 20H

يؤدي الى : -

- تخزين الرقم 80H في المسجل AL .
- راية الفيض في حالة 1 ، أي OF = 1

- راية الاشارة في حالة 1 ، أي SF = 1

والسبب في ذلك أن الرقم 8OH يمثل10000000B يكون سالبا فبدلا من 128+ حصلنا على - 128 . يحتاج الرقم 128+ الى 9 ثنائيات في حين ان المسجل AL يحتوي فقط على 8 ثنائيات . يمكن الحصول ايضا على حالة مشابهة لهذه الحالة عند جمع رقمين طول كل منهما كلمة واحدة والحصول على رقم اكبر من 32767 + . من أجل الحصول على جواب صحيح في مثل هذه الحالات يجب زيادة طول المكان المخصص لتخزين ناتج العملية . فمثلا في حالة جمع محتويات المسجل AL مع الرقم 20H ، يمكن نشر محتويات هذا المسجل الى المسجل AX باستخدام التعليمة CBW بالشكل التالي :



ومن ثم كتابة تعليمة ADD :

```
CBW
ADD AX , 20H
```

جمع الارقام متعددة البايت

ذكرنا سابقا ان التعليمات ADD و SUB تستخدم لمعالجة الارقام التي طولها بايت او كلمة.تحتاج معالجة الارقام التي يزيد طولها عن كلمة واحدة الى عمليات برمجة خاصة . لنفرض انه يراد جمع رقمين طول كل منهما 5 بايت وهما مخزنان في الذاكرة في العناوين FIRST و SECOND . ينفذ البرنامج التالي هذه العملية ويخزن المجموع في العنوان THIRD

```
;    MULTI-BYTE ADDITION   OF BIN   NUMBERS      *********
                     PAGE        60,60
CODE                 SEGMENT     PARA ' CODE '
                     ASSUME      CS:CODE, DS:CODE, SS: CODE
                     ORG         100H
BEGIN:            JMP            MAIN
;   *********       DATA    DEFINITION   *************
FIRST    DB          10H,20H,30H,40H,40H,50H
SECOND   DB          30H,25H,15H,8H,17H
THIRD    DB          6 DUP ( ? )
;     ******************************************
MAIN:    CLC
                     MOV         CX, 5          ; BYTE COUNTER
                     LEA         SI, FIRST
                     LEA         DI, SECOND
                     LEA         BX, THIRD
```

```
LL:
            MOV     AX, [ SI ]
            ADC     AX, [ DI ]
            MOV     [ BX ], AX
            INC     SI
            INC     DI
            INC     BX
            LOOP    LL
            RET
CODE        ENDS
            END     BEGIN
```

لنفرض ان الارقام (5040302010H) FIRST ، والرقم (1708152530H) SECOND مخزنة في الذاكرة الرئيسية بحيث يخزن البايت الاقل وزنا (Low order byte) في موقع الذاكرة الاقل عنوانا .

يوضح الشكل التالي كيفية تخزين الارقام في الذاكرة .

10H	FIRST
20H	
30H	
40H	
50H	
30H	SECOND
08H	
15H	
25H	
17H	
?	THIRD
?	
?	
?	
?	
?	

تخزين البيانات في الذاكرة قبل التنفيذ

10H	FIRST
20H	
30H	
40H	
50H	
30 H	SECOND
08 H	
15 H	
25 H	
17 H	
40H	THIRD
28H	
45H	
65H	
67H	

تخزين البيانات في الذاكرة بعد التنفيذ

يوضح البرنامج ان عملية جمع رقمين طول كل منهما يزيد عن 2 بايت تتم بتكرار تنفيذ التعليمة ADC التي تجمع في كل مرة بايت واحد من كل رقم . تبدأ العملية بجمع البايت الأيمن من الرقمين وتخزين الناتج في البايت الايمن من الحقل THIRD ، ثم يجمع البايت الثاني من اليمين من الرقمين مع القيمة الناتجة في راية الحمل وتخزين الناتج في البايت الثاني من الحقل THIRD . وهكذا تستمر هذه العملية حتى يتم جمع كل البايتات في الرقمين.تتطلب عملية الجمع هذه ما يلي : -

- تصفير راية الحمل قبل البدء بعملية الجمع .
- تخزين عدد البايتات في كل رقم في المسجل CX .

تمتاز هذه الطريقة بشمولها حيث يمكن استخدامها لجمع اي رقمين مهما كان طولهما . يلزم فقط تخزين طول الرقم في المسجل CX . عدا ذلك ، يمكن استخدام هذا البرنامج لطرح الارقام متعددة البايتات وذلك باستبدال التعليمة ADC بالتعليمة SBB.

جمع وطرح الارقام العشرية

ذكرنا سابقا ان الارقام العشرية تمثل في الصيغة المحزومة او المفكوكة . ان استخدام التعليماتSUB ,SBB , ADD , ADC على الارقام الممثلة في احدى هذه الاشكال تؤدي الى الحصول الى نتائج خاطئة . هذا لا يعني ان المعالج الدقيق لا يستطيع معالجة الارقام العشرية ، بل يلزم فقط تعديل الجواب بواسطة احدى التعليمات التالية :

- AAA تستخدم في حالة جمع الارقام العشرية المفكوكة .
- DAA تستخدم في حالة جمع الارقام العشرية المفكوكة .
- ASS تستخدم في حالة طرح الارقام العشرية المحزومة .
- DAS تستخدم في حالة طرح الارقام العشرية المحزومة .

تكتب هذه التعليمات بدون معاملات OPERANDS وتعمل على تعديل القيمة المخزنة في المسجل AL لذا يجب كتابة تعليمة التعديل المناسبة بعد تعليمات الجمع او الطرح مباشرة . نوضح بالتفصيل كيف تتم عملية التعديل .

اولا : تفحص التعليمة AAA النصف الايمن من المسجل AL فإذا احتوى على رقم يتراوح بين 0 , 9 تتم عملية تصفير النصف الايسر من AL ووضع الرايات AF , CF في حالة صفر . اذا كان النصف الايمن من AL يحتوي على قيمة أكبر من 9 او كانت الراية AF في حالة "1" تنفذ الخطوات التالية : -

- اضافة الرقم 6 الى المسجل AL .
- اضافة الرقم 1 الى المسجل AH .
- وضع الرايات AF , CF في حالة " 1 " .
- تصفير النصف الايسر من المسجل AL .

وبهذا تكون القيمة الناتجة في المسجل AL رقما عشريا مفكوكا صحيحا .

```
ADD  AL, BL
AAA
```

ثانيا : تعمل التعليمة DAA بطريقة مشابهة للتعليمة AAA ، غير ان التعليمة DAA تعالج الرقمين في المسجل AL : -

- فاذا كان النصف الايمن من المسجل AL يحمل قيمة أكبر من 9 أو AF = 1، يضاف الرقم 6 الى المسجل AL وتوضع الراية AF في حالة "1" .
- اذا كان النصف الايسر من المسجل AL أكبر من 9 او كانت الراية CF في حالة "1" تتم اضافة الرقم 60H الى المسجل AL وتوضع الراية CF في حالة "1"

وبهذا يحتوي المسجل AL دائما على رقمين عشريين صحيحين في الصيغة المحزومة .

```
ADD  AL , BL
DAA
```

ثالثا : تفحص التعليمة AAS النصف الايمن من المسجل AL ، فاذا احتوى على رقم يتراوح بين 0 , 9 تتم عملية تصفير النصف الايسر من المسجل AL وتوضع الرايات AF , CF في حالة "0" اذا كان النصف الايمن من AL يحتوي على قيمة أكبر من 9 او كانت الراية AF في حالة "1" تنفذ الخطوات التالية : -

- يطرح الرقم 6 من المسجل AL .
- يطرح الرقم 1 من المسجل AH .
- توضع الرايات AF , CF في حالة "1" .
- يصفّر النصف الايسر من المسجل AL .

وبهذا يحتوي المسجل AL على رقما عشريا صحيحا مفكوكا .

```
SUB   AL , BL
AAS
```

رابعا : تعمل التعليمة DAS بطريقة مشابهة للتعليمة AAS ، غير ان التعليمة DAS تعالج الرقمين في المسجل AL :

- اذا كان النصف الايمن من المسجل AL يحمل قيمة اكبر من 9 او كانت الراية AF في حالة "1" ، يطرح الرقم 6 من المسجل AL وتوضع الراية AF في حالة "1" .
- اذا كان النصف الايسر من المسجل AL يحمل قيمة أكبر من 9 او كانت الراية CF في حالة "1" يطرح الرقم 60H من المسجل AL وتوضع الراية AF في حالة "1" .

```
SUB   AL , BL
DAS
```

مثال ١ : اذا علمت ان : -

AL = 18 , BL = 54

اوجد محتويات المسجل AL بعد تنفيذ التعليمات :

ADD AL , BL
DAA

الحل : تقوم التعليمة ADD بجمع الرقمين وتخزين المجموع في المسجل AL :

6C = 54 + 18

نظرا لان المجموع يحتوي على قيمة ليست عشرية (C) ، لذا فان التعليمة DAA تقوم بتعديل المجموع بإضافة الرقم 6 الى المجموع 6C + 06 = 72

مثال ٢ : اذا علمت ان :

AL= 54 , BL = 18 اوجد محتويات المسجل AL بعد تنفيذ التعليمات :

SUB AL , BL

DAS

الحل : تقوم التعليمة SUB بطرح الرقمين وتخزين الناتج في المسجل AL

3C = 18 - 54

نظرا لان الناتج يحتوي على قيمة ليست عشرية (C) ، لذا فان التعليمة DAS تطرح الرقم 6 من الناتج : -

3C = 06 - 3C

مثال ٣ : البرنامج التالي يوضح عملية جمع الارقام المخزنة بنظام ASCII

```
;**********         ASCII    ADDITION     *********************
                     PAGE        60,60
CODE                 SEGMENT     PARA ' CODE '
                     ASSUME      CS:CODE, DS:CODE, SS:CODE
                     ORG         100H
BEGIN:   JMP            SHORT MAIN
;**********         DATA     DEFINITION     *******************
ASC1                 DB          '568'
ASC2                 DB          '695'
ASC3                 DB          '000'
;*****************************************************************
MAIN:
                     CLC
                     LEA         SI,ASC1+2
                     LEA         DI,ASC2+2
                     LEA         BX,ASC2+3
                     MOV         CX, 3          ;ASCII DIGIT COUNTER
```

```
LLL:
        MOV     AH, 00
        MOV     AL, [ SI ]
        ADC     AL, [ DI ]
        AAA                     ;ASCII ADJUST ADDITION
        MOV     [ BX ], AL
        DEC     SI
        DEC     DI
        DEC     BX
        LOOP    LLL
        MOV     [ BX ], AH
        RET
CODE    ENDS
        END     BEGIN
```

ضرب الارقام الثنائية

يحتوي طاقم تعليمات المعالج الدقيق 8086 / 8088 علي تعليمات خاصة لضرب الارقام . فالتعليمة MUL تستخدم لضرب الارقام غير المؤشرة . تستخدم التعليمة IMUL لضرب الارقام المؤشرة تستطيع كلتا التعليمتان ضرب رقمين طول كل منهما يساوي بايت واحد او كلمة واحدة . ففي حالة ضرب رقمين طول كل منهما يساوي بايت واحد يكون طول ناتج الضرب كلمة واحدة تؤدي عملية ضرب رقمين طول كل منهما كلمة (كلمة × كلمة) الى الحصول على ناتج بطول كلمتين .

بايت × بايت ⟵ كلمة واحدة (٢ بايت)

كلمة × كلمة ⟵ كلمتان (٤ بايت)

الصيغة العامة لتعليمات الضرب هي :

MUL Source - opernad

IMUL Source- opernad

حيث ان Source - opernad هو أحد معاملي عملية الضرب الذي يحدد طول الارقام المشاركة في العملية . فاذا كان طول source - opernad كلمة واحدة فالعملية هي كلمة × كلمة .

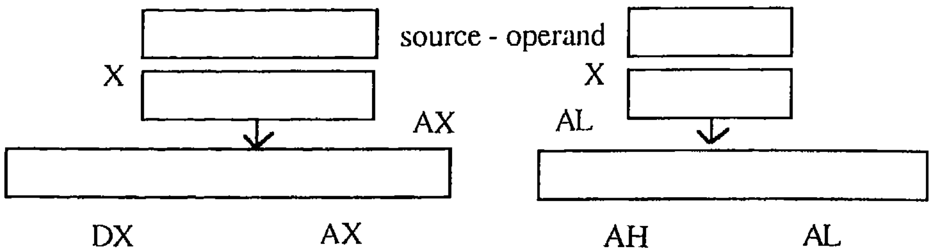
يخزن المعامل source - opernad في :

مسجل طوله بايت واحد من المسجلات عامة الاغراض أو في احد مواقع الذاكرة الذي طوله بايت واحد .

يخزن المعامل الثاني المشارك في العملية في المسجل AL في حالة ضرب بايت × بايت او في المسجل AX في حالة ضرب كلمة × كلمة . يخزن حاصل الضرب في المسجل

AX بعد ضرب بايت × بايت يخزن حاصل الضرب في المسجلين DX : AX في حالة ضرب كلمة × كلمة .

يبين الشكل الآتي معاملات عملية الضرب ومكان تخزين الناتج : -



ضرب البايت والكلمات

تؤثر التعليمات IMUL , MUL على راية الحمل CF وراية الفيض OF على النحو التالي :

- التعليمة MUL : توضع الرايات OF , CF في حالة الصفر اذا كان النصف الايسر لحاصل الضرب صفرا ، والا تكون الرايات في حالة الواحد.
- التعليمة IMUL : توضع الرايات OF , CF في حالة الصفر اذا كانت جميع الخانات في النصف الايسر لحاصل الضرب تحتوي صفرا او جميع الخانات في النصف الايسر لحاصل الضرب تحتوي واحدا . اي ان النصف الايسر لحاصل الضرب يكون امتدادا لإشارة الرقم في النصف الايمن . توضع الرايات OF , CF في الحالات الاخرى في حالة الواحد .

مثال : اكتب برنامج بلغة التجميع لضرب رقمين للحالات الآتية : -

- بايت × بايت
- كلمة × كلمة
- بايت × كلمة

بحيث تكون الارقام بدون إشارة او مع إشارة .

```
;;;    EXAMPLES OF MUL and IMUL   OPERATIONS ************
                    PAGE        60,60
CODE                SEGMENT     PARA ' CODE '
                    ASSUME      CS:CODE, DS:CODE, SS:CODE
                    ORG         100H
BEGIN:
                    JMP         MAIN
; **********DATA   DEFINITIONS      *********************
BYTE1    DB        80H
BYTE2    DB        40H
WORD1    DW        8000H
WORD2    DW        2000H
;  ********  UNSIGNED    MULTIPLICATION      MUL   *******
MAIN:
                    MOV         AL, BYTE1          ;BYET * BYTE
                    MUL         BYTE2
;
                    MOV         AX, WORD1          ;WORD * WORD
                    MUL         WORD2
;
                    MOV         AL, BYTE1          ;BYTE * WORD
                    SUB         AH, AH
                    MUL         WORD1
;   *******    SIGNED       MULTIPLICATION     IMUL      ******
                    MOV         AL, BYTE1          ;BYTE * BYTE
                    IMUL        BYTE2
;
                    MOV         AX, WORD1          ;WORD * WORD
                    IMUL        WORD2
;
                    MOV         AL, BYTE1          ;BYTE * WORD
                    CBW
                    IMUL        WORD1
;
                    RET
CODE                ENDS
                    END         BEGIN
```

يتضح من هذا المثال ان التعليمات IMUL , MUL تستخدم مباشرة لضرب الارقام التي لا يزيد طولها عن كلمة واحدة . تحتاج عملية ضرب الارقام التي يزيد طولها عن كلمة واحدة الى معالجة خاصة بحيث يقسم الرقمان الى اجزاء طول كل منها لا يزيد عن كلمة واحدة . تستخدم التعليمات MUL او IMUL لضرب البايتات او الكلمات واخيرا تتم عملية جمع حواصل الضرب والحصول على الجواب النهائي . لنفرض ان معالج دقيق يستطيع ضرب الارقام المكونة من منزلتين فقط ، فالعملية 1235 X 17 تتم على الشكل التالي : –

(1)	(2)	(3)
35	12	595
x17	x17	+20400
245	34	20995
35	17	
595	204	

يوضح المثال التالي عملية ضرب الارقام التي يزيد طولها عن كلمة واحدة ، يتكون المثال من جزئين : الاول يضرب كلمتان × كلمة ، الثاني يضرب كلمتان × كلمتان . تتم عملية الضرب في الجزء الاول

```
;    MULTI-WORD MULTIPLICATION    *********************
CODE             SEGMENT   PARA ' CODE '
                 ASSUME    CS: CODE, DS:CODE, SS: CODE
                 ORG       100H
BEGIN:  JMP        MAIN
;  ********  DATA DEFINITIONS       ******************
MULD             DW        3206H
                 DW        2521H
MULR             DW        6400H
                 DW        0A27H
PROD             DW        0
                 DW        0
                 DW        0
                 DW        0
;  *****   MULTIPLICATION OF  DOUBLEWORD * WORD   ****
MAIN:   MOV        AX, MULD+2  ;MULTIPLY RIGHT WORD
                 MUL       MULR          ;OF MULTIPLICAND
                 MOV       PROD+4, AX
                 MOV       PROD+2, DX
;
                 MOV       AX, MULD      ;MULTIPLY LEFT WORD
                 MUL       MULR          ;OF MULTIPLICAND
                 ADD       PROD+2, AX
                 ADC       PROD, DX
;
;  *****   MULTIPLICATION OF DOUBLEWORD * DOUBLEWORD
                 MOV       AX, MULD+2    ;MULTIPLICAND WORD-2
                 MUL       MULR+2        ; * MULTIPLIER WORD-2
                 MOV       PROD+6, AX
                 MOV       PROD+4, DX
;
                 MOV       AX, MULD+2    ;MULTIPLICAND WORD-2
                 MUL       MULR          ; * MULTIPLIER WORD-1
                 ADD       PROD+4, AX
                 ADC       PROD+2, DX
                 ADC       PROD, 00
;
                 MOV       AX, MULD      ; MULTIPLICAND WORD-2
                 MUL       MULR+2        ; * MULTIPLIER WORD-2
                 ADD       PROD+4,AX
                 ADC       PROD+2, DX
                 ADC       PROD, 00
;
                 MOV       AX, MULD      ;MULTIPLICAND WORD-1
                 MUL       MULR          ; * MULTIPLIER WORD-1
                 ADD       PROD+2, AX
                 ADC       PROD, DX
;
                 RET
CODE             ENDS
                 END       BEGIN
```

كما يلي

المعامل الاول WORD 1 WORD 2

المعامل الثاني MULR

1) WORD 2

x MULR

PROD + 2 PROD + 4

2) WORD 1

x MULR

DX AX

3) PROD + 2 PROD + 4

+ DX AX

PROD PROD + 2 PROD + 4

تنفذ عملية الضرب كلمتان x كلمتان على النحو التالي :-

المعامل الاول WORD1 WORD2

المعامل الثاني MULT1 MULT2 x

1) WORD 2

x MULT 2

PROD + 4 PROD + 6

2) WORD 2

x MULT 1

DX AX

3) PROD + 2 PROD + 4

+ DX AX

PROD + 2 PROD + 4

4) WORD 1

x MULT 2

DX AX

5)

	PROD + 2	PROD + 4
+	DX	AX
	PROD + 2	PROD + 4

6)

	WORD 1
	x MULT 1
DX	AX

7)

	PROD	PROD + 2
+	DX	AX
	PROD	PROD 2

وبهذا يكون حاصل الضرب النهائي في مواقع الذاكرة على الشكل التالي:

PROD   PROD +2 PROD + 4 PROD + 6

• يلزم استخدام التعليمة AAM لتعديل حاصل الضرب في حالة ضرب الارقام العشرية المفكوكة في حالة العملية بايت × بايت ، حيث تقوم التعليمة AAM بتحويل حاصل الضرب في المسجل AX الى الصيغة المفكوكة في المسجلات AL , AH . تتم عملية التحويل على النحو التالي : تقسم محتويات المسجل AL على الرقم 10 ويخزن ناتج القسمة في المسجل AH وباقي القسمة في المسجل AL . نفرض ان محتويات المسجلات قبل عملية الضرب هي : -

BL = 7 (00000111B) ,   AL = 9 (00001001B)

فبعد تنفيذ التعليمة MUL BL تصبح محتويات المسجلات هي :

BL = 7 (00000111B) , AH = 0   , AL = 63 (00111111B)

لتعديل حاصل الضرب الناتج في المسجلات AL ، AH تنفذ التعليمة AAM ، حيث تقسم محتويات المسجل AL (00111111B) على الرقم 10 وتخزن ناتج القسمة (00000110B) في المسجل AH وباقي القسمة (00000011B) في المسجل AL . وبالتالي فان حاصل الضرب 9 X 7 = 63 يصبح مخزنا في المسجل AX .

قسمة الأرقام الثنائية

تستخدم التعليمة DIV لقسمة الارقام غير المؤشرة . تستخدم IDIV لقسمة الارقام المؤشرة . هناك نوعان من القسمة : -

- قسمة كلمة على بايت : كلة ÷ بايت
- قسمة كلمتين على كلمة : كلمتان ÷كلمة

الصيغة العامة لتعليمات القسمة هي :

DIV divisor

IDIV divisor

حيث ان divisor المقسوم عليه هو الذي يحدد نوع عملية القسمة . فاذا كان طول المقسوم عليه بايت فالعملية هي كلمة ÷ بايت .

اما اذا كان طول المقسوم عليه كلمة فالعملية هي كلمتان ÷ كلمة.

يخزن المقسوم عليه اما في احد المسجلات عامة الاغراض او في احد مواقع الذاكرة .

يخزن المقسوم إما في المسجل AX في حالة قسمة كلمة ÷ بايت ، أو في المسجلين DX : AX في حالة قسمة كلمتان ÷ كلمة .

يكون ناتج القسمة اما في المسجل AL في حالة قسمة كلمتان ÷ بايت او في المسجل AX في حالة قسمة كلمتان ÷ كلمة . يخزن باقي القسمة في المسجل AH حالة كلمة ÷ بايت ، او في المسجل DX في حالة كلمتان ÷ كلمة .

يبين الشكل الاتي عملية القسمة :

AL	AX
Divisor ⟌ AX	Divisor ⟌ DX : AX
باقي القسمة في AH	باقي القسمة في DX

يوضح المثال التالي اشكال عملية القسمة المختلفة : -

- كلمة ÷ بايت
- بايت ÷ بايت
- كلمتان÷كلمة
- كلمة ÷ كلمة

للارقام غير المؤشرة والمؤشرة :

```
; *****   EXAMPLES OF DIVISION OPERATIONS  **************
                  PAGE       60,60
CODE              SEGMENT    PARA ' CODE '
                  ASSUME     CS:CODE, DS:CODE, SS:CODE
                  ORG        100H
BEGIN:
                  JMP        SHORT MAIN
; ********   DATA     DEFINITIONS        ****************
BYTE1    DB          80H
BYTE3    DB          16H
WORD1    DW          2000H
WORD2    DW          0010H
WORD3    DW          1000H
;    ***********************************************************
MAIN              PROC       NEAR
                  CALL       UNSDIV
                  CALL       SIGDIV
                  RET
MAIN              ENDP
;  ********  UNSIGNED DIVISION PROCEDURE    UNSDIV   *****
UNSDIV   PROC
                  MOV        AX, WORD1      ;WORD / BYTE
                  DIV        BYTE1

                  MOV        AL, BYTE1      ;BYTE / BYTE
                  SUB        AH, AH
                  DIV        BYTE3

                  MOV        DX, WORD2      ;DOUBLEWORD / WORD
                  MOV        AX, WORD3
                  DIV        WORD1

                  MOV        AX, WORD1      ;WORD / WORD
                  SUB        DX,DX
                  DIV        WORD3
                  RET
UNSDIV   ENDP
;  *******  SIGNED DIVISION PROCEDURE SIGDIV     *********
SIGDIV   PROC
                  MOV        AX, WORD1      ;WORD / BYTE
                  IDIV       BYTE1
;    ...............................................
                  MOV        AL, BYTE1      ;BYTE / BYTE
                  CBW
                  IDIV       BYTE3
;    ...............................................
                  MOV        DX, WORD2      ;DOUBLEWORD / WORD
                  MOV        AX, WORD3
                  IDIV       WORD1
;    ...............................................
                  MOV        AX, WORD       ;WORD / WORD
                  CWD
                  IDIV       WORD3
;    ...............................................
                  RET
SIGDIV   ENDP
CODE              ENDS
                  END        BEGIN
```

ضرب وقسمة الارقام الممثلة بنظام ASCII

يلزم في كثير من الاحيان إدخال المعطيات المراد معالجتها في البرنامج عن طريق لوحة المفاتيح . تمثل المعطيات المدخلة عن طريق لوحة المفاتيح في ذاكرة الحاسب بنظام ASCII . فمثلا يؤدي ادخال الرموز SAM من لوحة المفاتيح الى ادخال القيم 53414D . يؤدي ادخال الارقام 1234 الى تخزين القيم 31323334 . يتطلب تنفيذ العمليات الحسابية على البيانات المدخلة من لوحة المفاتيح الى تحويلها من نظام ASCII الى النظام الثنائي . يلزم بعد تنفيذ العمليات الحسابية والحصول على النتائج الى تحويلها من النظام الثنائي الى نظام ASCII . غير ان طاقم تعليمات المعالج الدقيق 8088 / 8086 يحتوي على تعليمات خاصة تسهّل اجراء العمليات الحسابية المختلفة على البيانات الممثلة بنظام ASCII مباشرة دون تحويلها الى النظام الثنائي . لقد تحدثنا سابقا عن تنفيذ عمليات الجمع والطرح على الارقام العشرية بالصيغة المفكوكة والمحزومة . نوضح الآن كيف تنفذ عمليات الضرب والقسمة على الارقام الممثلة في نظام ASCII . تتلخص عملية ضرب الارقام الممثلة في نظام ASCII في الخطوات الآتية : -

- تحوّل الارقام المشاركة في عملية الضرب الى الصيغة العشرية المفكوكة Un-packed Decimal

- تضرب الارقام باستخدام التعليمة MUL
- يعدّل حاصل الضرب باستخدام التعليمة AAM
- يحوّل حاصل الضرب الى نظام ASCII

لنفرض ان المسجل AL يحتوي 35H , CL يحتوي 39H تستخدم التعليمات التالية لضرب محتويات المسجلين وتحويل الناتج الى نظام ASCII .

```
AND   CL  , 0FH;  تحويل CL الى الرقم 9
AND   AL  , 0FH;  تحويل AL الى الرقم 5
MUL   CL  ; AL X CL        = 002DH
AAM               تحويل الناتج الى الصيغة المفكوكة
OR  AX, 3030H ; ASCII   تحويل الناتج الى نظام
```

يوضح البرنامج التالي ضرب رقم طوله 4 بايت برقم آخر طوله 1 بايت .

```
;   *******    ASCII    MULTIPLICATION             *************
CODE        SEGMENT    PARA 'CODE'
            ASSUME     CS:CODE,DS:CODE,SS:CODE
            ORG        100H
BEGIN:      JMP        MAIN
;----------------------------------------------------
MULD        DB         '3789'
MULR        DB         '6'
PROD        DB         5 DUP (0)
;----------------------------------------------------
MAIN:       MOV        CX,04
            LEA        SI,MULD+3
            LEA        DI,PROD+4
            AND        MULR,0FH
RRR:        MOV        AL,[SI]
            AND        AL,0FH
            MUL        MULR
            AAM
            MOV        [DI],AL
            DEC        SI
               DEC        DI
            LOOP       RRR
            RET
CODE        ENDS
            END        BEGIN
```

تتلخص عملية قسمة الارقام الممثلة في نظام ASCII بالخطوات التالية :

- تحويل المقسوم والمقسوم عليه الى الصيغة العشرية المفكوكة .
- تحويل المقسوم الى النظام الثنائي باستخدام AAD
- تقسم الارقام باستخدام DIV

لنفرض ان :

CL = 37 AX = 3238

تستخدم التعليمات التالية لقسمة الرقم 28 على 7

```
AND     CL  , 0 FH ;   الى 7        تحويل CL
AND     AX, OFOFH ;    الى 0208     تحويل AX
AAD                ;   التحويل الى النظام الثنائي
DIV     CL         ;   CL   على   AX   قسمة
```

يوضح البرنامج التالي قسمة رقم طوله 4 بايت على رقم طوله 1 بايت

```
;   ******    ASCII DIVISION          *****************
CODE        SEGMENT   PARA 'CODE'
            ASSUME    CS:CODE,DS:CODE,SS:CODE
            ORG       100H
BEGIN:      JMP       MAIN
;-----------------------------------------------
DIVD        DB        '3698'
DIVS        DB        '4'
QUOT        DB        4 DUP (0)
;-----------------------------------------------
MAIN:       MOV       CX,04
            SUB       AH,AH
            AND       DIVS,0FH
            LEA       SI,DIVD
            LEA       DI,QUOT
RRR:        MOV       AL,[SI]
            AND       AL,0FH
            AAD
            DIV       DIVS
            MOV       [DI],AL
            INC       SI
            INC       DI
            LOOP      RRR
            RET
CODE        ENDS
            END       BEGIN
```

تحويل الارقام من نظام ASCII الى النظامBCD

تمثل الارقام في ذاكرة الحاسوب في احد الانظمة التالية :

- النظام الثنائي
- نظام ASCII
- نظام BCD

فمثلا الرقم العشري 29 يمثل كما يلي :-

- في النظام الثنائي : 00011101
- في نظام ASCII : 0011001000111001
- في نظام BCD : 00101001

لقد صممت تعليمات العمليات الحسابية بحيث تنفذ مباشرة على الارقام الممثلة في النظام الثنائي مما يؤدي الى زيادة فعالية الحاسوب بشكل عام . غير انه يمكن اجراء العمليات الحسابية على الارقام الممثلة في نظام ASCII او نظام BCD . يتطلب ذلك اتخاذ اجراءات خاصة من قبل المبرمج لذا يلزم تحويل الارقام من نظام لآخر. فمثلا يستخدم البرنامج التالي لتحويل رقم في نظام ASCII طوله 3 كلمات الى نظام BCD

```
;    ******* ASCII TO BCD   CONVERSION    ****************
CODE        SEGMENT   PARA 'CODE'
            ASSUME    CS:CODE,DS:CODE,SS:CODE
            ORG       100H
BEGIN:      JMP       SHORT MAIN
;-----------------------------------------------------
ASC         DB.       '056328'
BCD         DB        '000'
;-----------------------------------------------------
MAIN        PROC      NEAR
            LEA       SI,ASC+4      ;INITIALIZE FOR ASCII
            LEA       DI,BCD+2
            CALL       ASCBCD
            RET
MAIN        ENDP
;-----------------------------------------------------
ASCBCD      PROC
            MOV        CL,04
            MOV        DX,03           ;NUMBER OF WORDS TO COURECT
XXX:        MOV        AX,[SI]
            XCHG       AH,AL
            SHL        AH,CL
            SHL        AX,CL
            MOV        [DI],AH
            DEC        SI
            DEC        SI
            DEC        DI
            DEC        DX
            JNZ        XXX
            RET
ASCBCD      ENDP
CODE        ENDS
            END       BEGIN
```

تتلخص خطوات تحويل الارقام من نظام ASCII الى نظام BCD فيما يلي : -

- تتم عملية التحويل كلمة بعد كلمة من اليمين الى اليسار .
- إزاحة البايت الايمن الى اليسار 4 مرات .
- إزاحة الكلمة الى اليسار 4 مرات .
- ينتج في البايت الايسر من الكلمة الرقم في نظام BCD الذي يخزن في المكان المخصص .
- الانتقال الى الكلمة التالية .

فمثلا يحول الرقم '3567' الممثل في نظام ASCII الى نظام BCD كما يلي : -

- يخزن الرقم "3567 في الذاكرة على الشكل 33353637
- يتكون الرقم من كلمتين : اليمنى 3637 واليسرى 3335 .
- تحول اولا الكلمة اليمنى 3637 .

- ازاحة البايت الايمن الى اليسار 4 مرات فتنتج 37 ←——— 3670
- ازاحة الكلمة الى اليسار 4 مرات فتنتج 6700 ←——— 3670
- يحتوي البايت الايسر الرقم 67 الذي يمثل الرقم 3637 في نظام BCD .

تحويل الارقام من نظام ASCII الى النظام الثنائي

يمكن تحويل الارقام الممثلة في نظام ASCII الى النظام الثنائي حسب الخطوات التالية : -

- تتم عملية التحويل بايت بعد بايت من اليمين الى اليسار .
- يصفر النصف الايسر من كل بايت .

يضرب البايت الاول بالرقم 1 ، البايت الثاني بالرقم 10 ، البايت الثالث بالرقم 1000 وهكذا .

- تجمع حواصل الضرب مع بعضها .

فمثلا يحول الرقم '1234' الممثل في نظام ASCII الى النظام الثنائي كما يلي :-

السادس عشري	النظام العشري	
4	4	4 x 1 =
1E	30	3 x 10 =
C8	200	2 x 100 =
3E8	1000	1 x 1 000 =
04D2		المجموع

تحويل الارقام من النظام الثنائي الى نظام ASCII

تخزن نتائج العمليات الحسابية في الذاكرة في النظام الثنائي. تقوم الشاشة او الالة الطابعة بعرض البيانات الممثلة في نظام ASCII لذا يجب تحويل البيانات الى نظام ASCII قبل عرضها . يحول الرقم الثنائي الى نظام ASCII وذلك بقسمته على الرقم 10 وحساب ناتج وباقي القسمة. ثم يقسم ناتج القسمة على الرقم 10 ويحسب الناتج والباقي ثانية وهكذا تستمر العملية حتى يصبح ناتج القسمة اقل من 10 . وبهذا يشكل الرقم المكوّن من بواقي القسمة الرقم في نظام ASCII بعد اضافة الرقم 3 الى يسار كل من بواقي القسمة. فمثلا يحول الرقم 04D2 الى نظام ASCII كما يلي :-

	ناتج القسمه	باقي القسمه
04D2 ÷ A =	7B	4
7B ÷ A =	C	3
C ÷ A =	1	2

يكون الجواب 1234 الذي يخزن في الذاكرة كما يلي : -

31323334

يوضح البرنامج التالي عمليات التحويل من نظام ASCII الى النظام الثنائي وبالعكس .

```
;    ****** ASCII TO BINARY    &  BINARY TO ASCII  CONVERSION
CODE        SEGMENT   PARA 'CODE'
            ASSUME    CS:CODE,DS:CODE,SS:CODE
            ORG       100H
BEGIN:      JMP       SHORT MAIN
;-------------------------------------------
ASC         DB        '1234'
BIN         DW        0
ASCLEN       DW        4
MULT10       DW        1
;-------------------------------------------
MAIN        PROC      NEAR
            CALL       ASCBIN
            CALL      BINASC
            RET
MAIN        ENDP
;-------------------------------------------
;     CONVERT    ASCII TO BINARY
ASCBIN      PROC
            MOV        CX,10       ;MULTIPLY FACTOR
            LEA       SI,ASC-1    ;ADDRESS OF ASC
            MOV       BX,ASCLEN
XXX:
            MOV       AL,[SI+BX]
            AND       AX,000FH
            MUL       MULT10
            ADD       BIN,AX
            MOV       AX,MULT10
            MUL       CX
            MOV       MULT10,AX
            DEC       BX
            JNZ       XXX
            RET
ASCBIN      ENDP
;      CONVERT BINARY TO ASCII
BINASC      PROC
            MOV       CX,0010     ;DIVIDE FACTOR
            LEA       SI,ASC+3    ;ADDRESS OF ASC
            MOV       AX,BIN
```

```
RRR:
           CMP        AX,0010       ;BIN <10 ?
           JB         BELOW         ;YES -EXIT
           XOR        DX,DX
           DIV        CX             ;DIVID BY 10
           OR         DL,30H
           MOV        [SI],DL        ;STORE ASCII CHAR
           DEC        SI
           JMP        RRR
BELOW:     OR         AL,30H
           MOV        [SI],AL
           RET
BINASC     ENDP
CODE       ENDS
           END        BEGIN
```

مثال : اكتب برنامج لتحويل رقم من النظام السادس عشري مخزن في المسجل AL الى

- نظام ASCII
- نظام BCD
- نظام EBCDIC

```
;  **** HEX  TO ASCII      &
;  **** HEX  TO BCD        &
;  **** HEX  TO EBCDIC     ,  CONVERSIONS
CODE       SEGMENT   PARA 'CODE'
           ASSUME    CS:CODE,DS:CODE,SS:CODE
           ORG       100H
           MOV       AL,9H
BEGIN:     JMP       MAIN
;--------------------------------------------
ASCII      DB        '0123456789ABCDEF'
BCD        DB        0,1,2,3,4,5,6,7,8,9,10H,11H,12H,13H,14H,15H
EBCDIC     DB        0F0H,0F1H,0F2H,0F3H,0F4H,0F5H,0F6H
           DB        0F7H,0F8H,0F9H,0C1H,0C2H,0C3H,0C4H
           DB        0C5H,0C6H
;--------------------------------------------
MAIN       PROC      NEAR
           MOV       AL,9H
           CALL      CONVER
           RET
MAIN       ENDP
;--------------------------------------------
CONVER     PROC
           MOV       DL,AL
           LEA       BX,ASCII
           XLAT      ASCII
           MOV       CH,AL       ;STORE IN CH
           MOV       AL,DL
           LEA       BX,BCD
           XLAT      BCD
           MOV       CL,AL       ;STORE IN CL
           MOV       AL,DL
           LEA       BX,EBCDIC
           XLAT      EBCDIC
           MOV       AH,AL       ;STOR IN AH
           RET
CONVER     ENDP
CODE       ENDS
           END       BEGIN
```

الوحدة السادسة

نقل التحكم والاوامر المنطقية

- تعليمات القفز ، الاستدعاء
- تعليمات نقل التحكم المشروطة
- تعليمات الاعتراض
- التعليمات المنطقية

نقل التحكم والاوامر المنطقية

CONTROL TRANSFERING AND PROGRAM LOGIC

يتم تنفيذ البرنامج عادة بشكل تسلسلي وفي بعض الاحيان قد يتطلب الامر نقل التحكم (التنفيذ) الى تعليمة معينة وتمتلك لغة التجميع مجموعة من التعليمات المخصصة لنقل التحكم والتي يمكن تصنيفها الى ما يلي :

١ – تعليمات القفز ، الاستدعاء والعودة (Jump , Call , Return)

٢ – تعليمات نقل التحكم المشروطة (Conditional transfer)

٣ – تعليمات التكرار (Iteration control)

٤ – تعليمات الاعتراض (Interrupts)

القفز الغير مشروط
unconditional Jumbs

تستخدم تعليمة القفز الغير مشروط (jumb) لنقل التحكم الى تعليمة لا تلي التعليمة التي قيد التنفيذ وقد يكون القفز اماميا (forward) او خلفيا (backward) وفي كلا الحالتين يتم حساب مقدار الازاحة في العنوان (بين عنوان التعليمة المنفذة والتعليمة التي سينتقل اليها التحكم) حيث تضاف هذه الازاحة الى محتوى مؤشر التعليمة (IP) . وتأخذ تعليمة القفز الشكل التالي

JMP Lable

حيث تأخذ العلامة اسما (ينطبق على اسم العلامة ما ينطبق على كافة الاسماء في لغة التجميع) ويكون متبوعا بالنقطتين اذا كانت العلامة اسما لفقرة داخل الاجراء (PROCEDURE) نفسه كما هو موضح في المثال التالي

```
MAIN:   PROC   NEAR
        MOV    AX , 01
        MOV    BX , 08
Z01:
        ADD    AX , 01
        ADD    BX , AX
        SHL    CX , 1
        JMP    Z01
```

اما اذا كان اسم الفقرة المراد نقل التحكم اليها خارج الاجراء فلا داعي لاستخدام النقطتين بعد اسم هذه الفقرة لاحظ في المثال الاسبق ان القفز هو خلفي وان قيمة الازاحة (-٩) والتي سوف تضاف الى قيمة العنوان المشار اليه في مؤشر التعليمة وذلك لنقل التحكم الى التعليمة (ADD AX, 01)

وعند التعامل مع تعليمة القفز الغير مشروط يجب مراعاة ما يلي:

١ - القفز القصير (SHORT Jump) حيث يتم تخصيص بايت واحد لمعامل التعليمه واكبر ازاحة للامام ستكون بمقدار ١٢٧ بايتا اما الازاحة للخلف (السالبة) فتكون (- ١٢٨) بايتا .

٢ - القفز الطويل (Far jump) حيث يتم تخصيص ٢ بايت لمعامل التعليمة (2 operands) اضافة الى بايت واحد لشيفرة التعليمه

وعند التعامل مع الازاحة الامامية يمكن الاعلان عن نوع القفز وإلا فإن الكمبيوتر سوف يعتبر القفز طويلا وذلك بتوليد ٣ بايت لهذه التعليمه (2 operands) ولتلافي هذا يمكن استخدام الامر (Short) في تعليمه القفز فمثلاً

```
        JMP  Z20
    Z20:
          :
```

يمكن كتابتها كما يلي :

```
      JMP SHORT Z20
   Z20:
          .
          .
          .
```

الاستدعاء والعودة
CALL & RETURN

تسهيلا لكتابة البرنامج ومراجعته ونظرا للحاجة الى تنظيم البرنامج وتقسيمه الى اجزاء (اجراءات) تحتوي على عمليات مخصصة لمعالجة معينة فان المبرمج يضطر الى تجزئة مقطع العمليات (code segment) الى اجراءات يتم استدعاؤها عند الحاجة اليها ومن اهم الامور الواجب مراعاتها عند التعامل مع الاجراءات ما يلي:

١ – مدخل البرنامج التنفيذي يحدد باستخدام المعامل FAR والذي يخبر الكمبيوتر بالبداية الفعلية للتعليمات.

٢ – المعامل NEAR يعني ان الاجراء هو فرعي وهو ضمن مقطع التعليمات.

٣ – يعطى كل اجراء اسما ويجب ان ينتهي الاجراء بالامر END

٤ – يتم استدعاء الاجراء باستخدام الامر CALL وبعد نقل التحكم الى الاجراء وتنفيذه تتم العودة الى التعليمة التي تلي CALL وذلك باستخدام الامر CALL .

من خلال الشكل التالي يمكن ايضاح كيفية استدعاء البرنامج الفرعي

```
CODESG    SEGMENT    PARA
BEGIN     PROC       FAR
          .
          .
          .
          CALL       PR1
          CALL       PR2
          RET
BEGIN     ENDP
PR1       PROC       NEAR
          .
          .
          .
          RET
PR1       ENDP
PR2       PROC       NEAR
          .
          .
          .
          RET

PR2       ENDP
CODESG    ENDS
          END        BEGIN
```

يحتوي مقطع التعليمات هذا على برنامجين فرعين هما PR2 , PR1 بحيث يتم استدعاءPR1 اولا باستخدام الامر CALL PR1 حيث ينتقل التحكم الى البرنامج الفرعي وعند مصادفة الامر RET يتم الانتقال الى الجملة التي تلي جملة PR1 CALL . (CALL PR2)

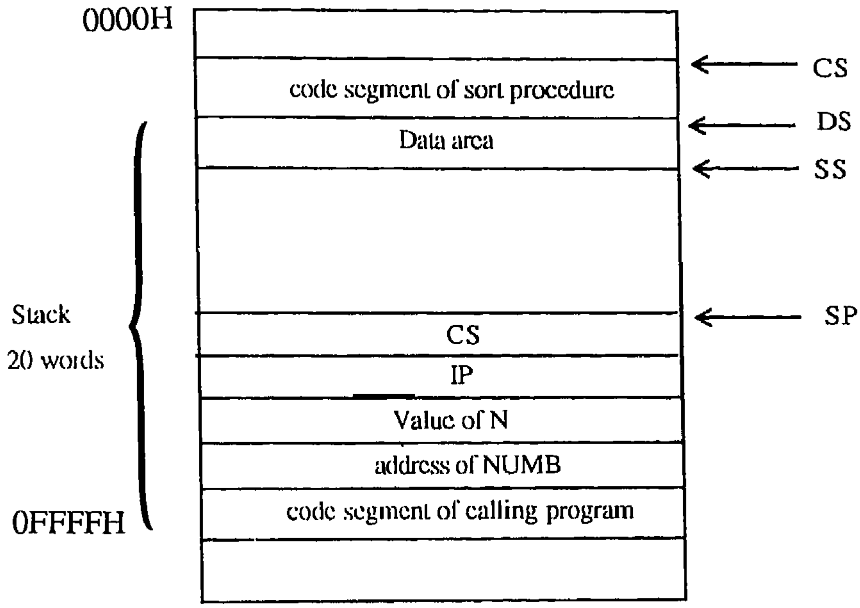
عند مصادفة الامر CALL فانه يتم الاحتفاظ بعنوان الامر التالي (محتوى IP) في الحزمة وذلك بدفعه(PUSH) في الحزمة اما الامر RET فيعني الرجوع الى الحزمة واسترجاع (POP) آخر عنوان مخزن بها .

ومن خلال البرنامج سوف نستعرض كيفية التعامل مع الحزمة وكيفية استدعاء البرنامج (اوامر القفز المشروطة سيتم استعراضها في هذا الباب) .

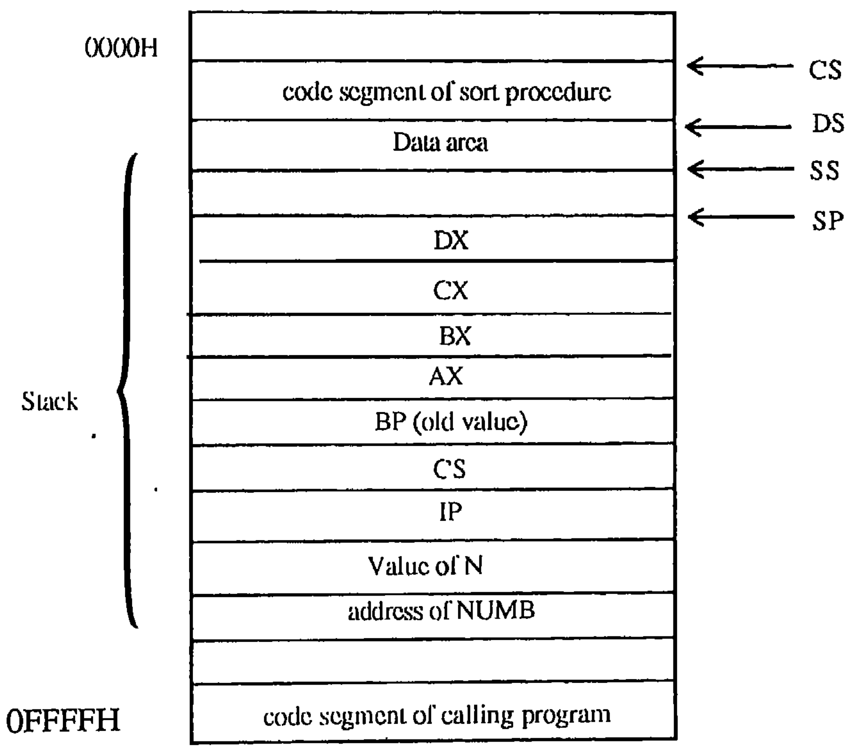
```
                PAGE 30.50
; THIS PROGRAM TO SORT A GIVEN ARRAY OF N BTYES
; INTO DESCENDING ORDER
STACK-AREA      SEGMENT
                DW 20 DUP(?)      ;SPACE FOR STACK
                TOS LABEL WORD    ;OFFSET OF TOP STACK
STACK-AREA      ENDS
SORT-DATA       SEGMENT
                NUMB DB 20,5,9,12,19   ;ARRAY TO BE SORTED
                N    DW  5            ;NUMBER OF ELEMENTS IN ARRAY
SORT-DATA       ENDS
SORT-EXAMPLE    SEGMENT
                ASSUME CS:SORT-EXAMPLE, DS:SORT-DATA,
                       SS:STACK-AREA
;
; SORT PROCEDURE
; BEGIN BY SAVING REGISTERS TO BE USED,ON THE STACK
SORT            PROC     FAR
                PUSH     BP    ;SAVE BASE POINTER
                MOV      BP,SP  ;GET CURRENT SP INTO BP
                PUSH     AX
                PUSH     BX
                PUSH     CX
                PUSH     DX
                MOV      DH,[BP+6]  ;GET VALUE IN DH
                                ;THIS IS ALSO THE CURRENT
                                  ;VALUE OF LAST
                MOV      BX,[BP+8]  ;GET ADDRESS OF FIRST ELEMENT
                                  ;OF ARRAY INTO BX
                ;INITIALIZE EXCH(AH REGISTER) AND NEXT(CL)
;
             MOV         AH,0
                MOV      CL,1
;    WHILE NEXT < LAST DO
; SEARCH FOR BEGER AND PERFORM SWAPPING
SCANBEG:        CMP      DH,CL     ;NEXT < LAST
                JL       SCANOVER ;IF NO SKIP SWAPPING
```

```
;           SWAPPING
                    MOV      AL,[BX]   ;ELEM1   = NUMB
                    CMP      AL,[BX+1] ;(NEXT)
                    JNL      NEXTCOMP  ;IF NO THEN DO NOT EXCHANGE
                    MOV      DL,AL     ;ELSE TEMP=NUMB(NEXT)
                    MOV      AL,[BX+1] ;NUMB(NEXT)=NUMB(NEXT+1)
                    MOV      [BX+1],DL ;NUMB(NEXT+1)=TEMP
                    MOV      AH,1      ;EXCHG=1 SWAPPING FLAG
;
NEXTCOMP:
                    INC      CL
                    JMP      SCANBEG
;
SCANOVER:
                    CMP      AH,0   ;IF EXCHG=0 THEN SORT OVER
                    JE       DONE
                    MOV      AH,0
                    MOV      CL,1
                    DEC      DH
                    JMP      SCANBEG
;
DONE:
                    POP      DX
                    POP      CX
                    POP      BX
                    POP      AX
                    POP      BP
                    RET      4        ;POP OUT THE 4 BYTES OF PARAMETERS
                    SORT     ENDP
                    SORT-EXAMPLE ENDS
CALL-SEG            SEGMENT
                    ASSUME CS:CALL-SEG, DS:SORT-DATA
                           SS:STACK-AREA
BEGIN:
                    MOV      AX,SORT-DATA
                    MOV      DS,AX
                    MOV      AX,STACK-AREA
                    MOV      SS,AX
                    MOV      SP,OFFSET TOS
                    MOV      AX,OFFSET NUMB
                    PUSH     AX
                    MOV      AX,N
                    PUSH     AX
                    CALL     SORT
                    CALL-SEG ENDS
                    END      BEGIN
```

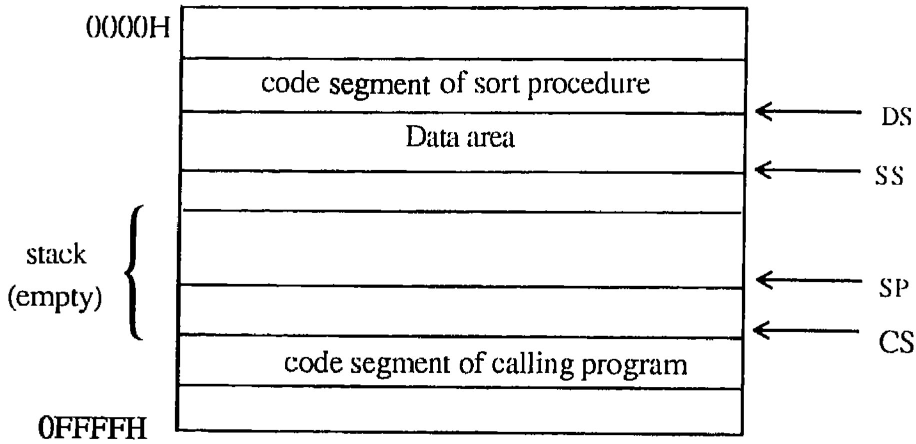
من خلال استعراض هذا البرنامج فإن شكل الذاكرة بعد عملية استدعاء برنامج الفرز يصبح كما يلي



وعند حفظ المسجلات في الحزمة فإن شكل الذاكرة بعد عملية الحفظ هذه سوف يصبح كما يلي :



وبعد العودة (RET) من برنامج الفرز فإن شكل الذاكرة يصبح كما يلي وذلك بعد استرجاع (popping) العناوين المخزنة بالحزمة



تعليمات نقل التحكم المشروطة
Conditional Jump INSTRUCTIONS

تستخدم هذه التعليمات لنقل التحكم الى تعليمة معينة اعتمادا على تحقق شرط وذلك بفحص الخلية المخصصة في المسجل FR

Bit no :	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Flag :	*	*	*	*	O	D	I	T	S	Z	*	A	*	P	*	C

ويتم احتساب مقدار الازاحة (كما في تعليمة نقل التحكم الغير مشروط) حيث تضاف هذه الازاحة الى محتوى مؤشر التعليمة . وهناك عدة انواع من هذه التعليمات نورد منها ما يلي :

١ - تعليمات القفز اعتمادا على استخدام بيانات بدون اشارة حيث تخصص كافة الخلايا الثنائية لتمثيل البيانات وتستخدم هنا . مجموعة من التعليمات هي

JE / JZ	jump equal or jump zero	tests	ZF
JNE / JNZ	jump not equal or not zero	=	ZF
JA / JNBE	jump above or not below / equal	=	CF , ZF

JAE / JNB	jump above / equal or not below	=	CF
JB / JNAE	jump below or not above / equal	=	CF
JBE / JNA	jump below / equal or not above	=	CF , AF

٢ – تعليمات القفز باستخدام بيانات رقمية مؤشرة (اعتمادا على الاشارة) حيث تخصص الخلية في اقصى اليسار من خلايا البيانات لتمثيل الاشارة (٥ للموجب ١ للسالب) وتستخدم هنا مجموعة من التعليمات منها :

JE / JZ	jump equal or zero	ZF
JNE / JNZ	jump not equal or not zero	ZF
JG/JNLE	jump greater or not less/equal	ZF, SF, OF
JGE / JNLE	jump greater / equal or not less	SF , OF
JL / JNGE	jump less or not greater / equal	SF , OF
JLE / JNG	jump less / equal or not greater	ZF , SF , OF

٣ – تعليمات الفحص الحسابي (Arithmetic test) ومن اهم هذه التعليمات ما يلي

JS	jump sign (negative	tests	SF
JNS	jump no sign (positive)	=	SF
JC	jump carry (same as JB)	=	CF
JNC	jump no carry	=	CF
JO	jump overflow	=	OF
JNO	jump no overflow	=	OF
JP / JPE	jump parity even	=	PF
JNP / JP	jump parity odd	=	PF

وعادة ما تستخدم هذه التعليمات لفحص خلية معينة (FLAG) بعد اجراء عملية مقارنة معينة كما يلي

```
CMP   BX, 00 ;  Compare  BX to zero
JZ     PARI   ; jump if zero to PARI
 .
 .     (action if nonzero)
 .
PARI : ...
       (action if zero)
```

لاحظ انه اذا كانت قيمة (BX) مساوية للصفر فإن قيمة ZF سوف تصبح مساوية للواحد وإلا فإن قيمتها ستصبح صفراً .

وهناك امر آخر يتم من خلاله التحقق من قيمة المسجل (CF) بدون التأثير على قيمة ZF ويتم نقل التحكم الى فقرة معينة عندما تصبح قيمة هذا المسجل مساوية للصفر .

JCXZ PARNI

وفي المثال الاسبق فإن عملية المبادلة

```
MOV     AL , [BX]
CMP     AL , [BX + 1]
JNL     NEXTCOMP
MOV     DL , AL
MOV     AL , [BX + 1]
MOV     [BX + 1] , DL
```

تتم عملية مقارنة محتوى BX مع محتوى BX + 1 فاذا كانت قيمة BX + 1 ليست اقل (Not Less) فإنه يتم الانتقال الى NEXTCOMP والا فإن عملية المبادلة سوف تنفذ .

التكرار (LOOP)

تختلف تعليمات التكرار عن تعليمات القفز في ان تنفيذ تعليمة او مجموعة من التعليمات قد يتكرر لحين تحقق شرط معين . وعند التعامل مع التكرار يجب مراعاة ما يلي .

- اعطاء المسجل CX قيمة ابتدائية مساوية لعدد المرات المراد تكرارها .
- عند الانتقال الى تنفيذ تعليمات التكرار فإن قيمة CX سوف تنقص اوتوماتيكيا بمقدار واحد .
- يبقى التكرار قائما ما دامت قيمة CX غير مساوية للصفر .

فمثلا لاحتساب مجموع الارقام من ١ الى ١٠ يمكن تنفيذ التكرار التالي

```
        MOV   BX ,00
        MOV   CX , 10
        MOV   AX , 00
S10:
       INC    AX
       ADD   BX , AX
       LOOP S10
```

لاحظ أنه عندما تصبح قيمة CX مساوية للصفر فإنه سوف يتم تنفيذ الجملة التي تلي جملة LOOP

وقد تستخدم تعليمة LOOP في عدة اشكال اهمها .

١ - LOOPE (LOOP equal)

٢ - LOOPNE (LOOP not equal)

حيث تقوم هذه التعليمات بانقاص قيمة CX وتكرار الدوارة (LOOP) ما دامت CX غير مساوية للصفر لكن التعليمة الاولى تبقي ZF = 1 اما الثانية فإن ZF = 0

وفي المثال التالي سوف نبين الفرق بين التكرار وتعليمات القفز

```
                PAGE 50,30
TITLE   BK3.ASM  ILLUSTRATION OF CALL AND JUMP
;================================================
STACKSG SEGMENT  PARA STACK 'STACK'
                DW      32 DUP(?)
STACKSG ENDS
;================================================
DATASG  SEGMENT  PARA 'DATA'
ZFLD1   DB      'ZIAD****'
FLD2            DB      'RASHAD**'
FLD3            DB      'FATTAH**'
DATASG  ENDS
;================================================
CODESG  SEGMENT PARA 'CODE'
BEGIN   PROC    FAR
                ASSUME  CS:CODESG;DS:DATASG,SS:STAKSG,ES:DATASG
;     PROGRAM INITIALIZATION .....................
;     ......................
                PUSH    DS
                SUB     AX,AX
                PUSH    AX
                MOV     AX,DS
;==========================
                MOV     DS,AX
                MOV     ES,AX
                CALL    PR1             ;CALL JUMP ROUTINE
                CALL    PR2             ;CALL LOOP ROUTINE
                RET
BEGIN   ENDP
;..... USING JUMP-ON-CONDITION .....................
;      ......................
PR1             PROC
                LEA     SI,ZFLD1
                LEA     DI,FLD2  ;INITIALIZE ADDRESS OF FLD1 AND FLD2
                MOV     CX,08    ;INITIALIZE COUNTER
ZZ1:
                MOV     AL,[SI]  ;MOVE FROM FLD1
                MOV     [DI],AL  ;MOVE TO FLD2
                INC     SI
                INC     DI
                DEC     CX
                JNZ     ZZ1     ;IF COUNTER=0 RETURN TO CALLER
                           ;OTHERWISE GOTO ZZ1
                RET
                ENDP
```

```
PR1
;==========================================================
;.....    USING LOOP .....................................
;         ..........
PR2              PROC
                 LEA      SI,FLD2
                 LEA      DI,FLD3
                 MOV      CX,08
ZZ2:
                 MOV      AL,[SI]
                 MOV      [DI],AL
                 INC      SI
                 INC      DI
                 LOOP     ZZ2        ;DEC COUNTER IF COUNTER=0
                                  ;THEN RETURN ELSE GO BACK
                                  ;TO ZZ2 (LOOP)
                 RET
PR2              ENDP
CODESG   ENDS
                 END      BEGIN
```

تعليمة الاعتراض Interrupt instruction INT

قد يتطلب الامر اثناء تنفيذ البرنامج اللجوء الى نظم التشغيل او اي برمجيات الادخال والاخراج وذلك لتنفيذ روتين معين لذا يتم بواسطة تعليمة الاعتراض قطع تنفيذ البرنامج وبعد تنفيذ الروتين المطلوب يتم الرجوع الى التعليمة التالية في البرنامج ويمكن ايجاز عمل هذه التعليمة فيما يلي : –

- انقاص قيمة SP بمقدار ٢ ودفع FR في الحزمة لحفظه لغاية الرجوع اليه لاحقا (حفظ حالة البرنامج) .
- تصفير IF , TF
- الاحتفاظ بقيمة CX وذلك بانقاص SP بمقدار ٢ ودفع CX .
- انقاص SP بمقدار ٢ ودفع IP
- تنفيذ الروتين المطلوب .
- العودة بعد التنفيذ الى متابعة تنفيذ البرنامج بدأ من التعليمة التي تلي تعليمة الاعتراض .

هذا وسوف نتناول عمليات الاعتراض بشيء من التفصيل عند الكلام عن برمجة الادخال والاخراج .

التعليمات المنطقية
Boolean instructions

للتعليمات المنطقية اهمية كبيرة في برنامج التجميع حيث يمكن بواسطتها تصفير بت معين او فحص بت ما ومن اهم هذه التعليمات نورد ما يلي

– تعليمة AND حيث تقوم هذه التعليمة بضرب محتوى معاملين منطقيا وذلك بضرب كل خلية ثنائية من المعامل الاول مع الخلية المناظرة في المعامل الثاني كما يلي

0111
0101
0101

– تعليمةOR حيث تقوم هذه التعليمة بجمع محتوى معاملين بحيث يتم جمع كل خلية ثنائية منطقيا من المعامل الاول مع الخلية المناظرة في المعامل الثاني

0111
0101
0111

– تعليمة XOR

تقوم هذه التعليمة بجمع خلايا المعامل الاول والثاني وتكون نتيجة الجمع مساوية للواحد اذا اختلفت قيم هذه الخلايا.

0111
0101
0010

– التعليمة TEST تشبه هذه التعليمة تعليمة AND وتقوم بالتأثير على FR ولكن محتوى المعاملات لا يتغير .

والامثلة التالية توضح مفهوم هذه التعليمات

لنفرض ان AL يحتوي على 1100 0101

وان BH يحتوي على 0101 1100

فان

AND	AL , BH	sets AL to 0100 0100
OR	BH , AL	sets BH to 1101 1101
XOR	AL , AL	sets AL to 0000 0000
AND	AL , OFH	sets AL to 0000 0101
OR	CL , CL	sets SF and ZF
TEST	AL, 00000001B	Does AL contain a odd number
TEST	DX, OFFH	Does Dx contain zero

وللتعليمات المنطقية اهمية في عملية التحويل بين الاحرف الكبيرة والصغيرة فالاحرف الكبيرة تبدأ من (41H) منتهية (5A) اما الصغرى فتبدأ من (61H) منتهية بالرقم (7A)

Bit 76543210

letter A : 01000001

letter Z : 01011010

Bit 76543210

letter a : 01100001

letter z : 01111010

فلو نفذ الأمر

AND AX , 11011111 B

فإن هذا سوف يؤثر فقط على البت رقم (5) بمعنى آخر في الاحرف الكبرى يبقى ثابتاً في حين تتغير قيمة هذا البت في الاحرف الصغرى وبالتالي يتم تغيرها الى أحرف كبيرة

والمثال التالي يوضح كيفية تحويل الاحرف الصغرى الى كبرى

```
                    PAGE 30,60
; PROGRAM TO CHANGE LOWERCASE TO UPPERCASE
CODESG  SEGMENT PARA 'CODE'
                    ASSUME  CS:CODESG,DS:CODESG,SS:CODESG
                    ORG     100H
BEGIN:          JMP         MAIN
TITLEX  DB          'change to uppercase letters'
MAIN    PROC        NEAR
                    LEA     BX,TITLEX+1     ;FIRST CHAR TO CHANGE
                    MOV     CX,31           ;NO. CHAR TO CHANGE
A20:
                    MOV     AH,[BX]         ;CHAR FROM TITLEX
                    CMP     AH,61H          ;IS IT
                    JB      B30             ;LOWER
                    CMP     AH,7AH          ;CASE
                    JA      B30             ;LETTER ?
                    AND     AH,11011111B    ;YES CONVERT
                    MOV     [BX],AH
B30:
                    INC     BX
                    LOOP    A20
                    RET
MAIN    ENDP
CODESG  ENDS
                    END     BEGIN
```

الازاحة والدوران
SHIFTING AND ROTATING

تقوم تعليمات الازاحة بازاحة الخلايا الثنائية بحيث يتم الاحتفاظ بالخلية (bit) الخارجة في CF ومن هذه التعليمات :

١ – الازاحة بدون الاشارة لليمين SHR

٢ – الازاحة بدون الاشارة لليسار SHL

٣ – الازاحة الحسابية لليمين SAR

٤ – الازاحة الحسابية لليسار SAL

تتم عملية الازاحة بمقدار خلية واحدة وعند اجراء عملية الازاحة لاكثر من خلية يتم استخدام CL وذلك لتخزين قيمة الازاحة فيه والامثلة التالية توضح مفهوم الازاحة :

```
MOV   CL , 03           ; AX
MOV   AX, 10110111B     ; 10110111
SHR   AX , 1            ; 01011011
SHR   AX , CL           ; 00001011
```

لاحظ انه يتم اضافة صفر من اليسار اما قيمة البت الخارج من اقصى اليمين (SHR) فيتم الاحتفاظ به في CF . اما التعليمة SAR (SAL) فتختلف عن تعليمة الازاحة (SHR) في انه يتم تكرار الاشارة في الخانات اليسرى والمثال التالي يوضح هذا

```
MOV   CL ,03            ;  AX:
MOV   AX , 10110111 B   ; 10110111
SAR   AX, 1             ; 11011011
SAR   AX , CL           ; 11111011
```

كما ويمكن استخدام تعليمات الازاحة لليسار واليمين لمضاعفة الرقم او قسمته على ٢ بدلا من استخدام تعليمات الضرب او القسمة .

اما تعليمات الدوران فتتم باحداث دائرة مقفلة من الازاحة ومن هذه التعليمات

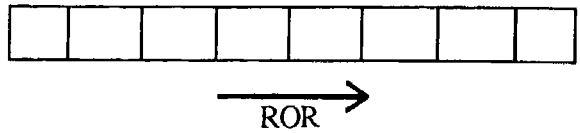
١ – دوران لليمين ROR

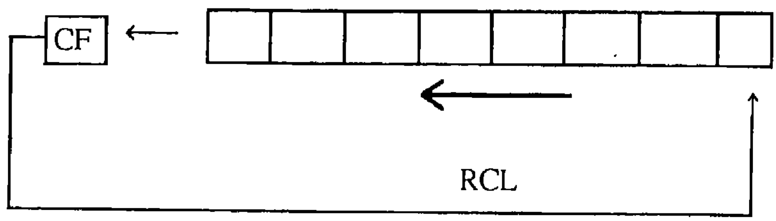
٢ – دوران لليسار ROL

٣ – دوران مع الحمل لليمين RCR حيث يشترك CF في حلقة الدوران .

٤ - دوران مع الحمل لليسار RCL

والاشكال التالية توضح مفهوم هذه التعليمات





فمثلا تنفيذ التعليمات التالية

```
SHL  AX , 1
RCL  DX , 1
```

يؤدي الى ضرب محتوى كل من DX , AX في (٢)

وفيما يلي برنامجين يوضحان كيفية استخدام التعليمات المنطقية وغيرها من التعليمات التي تم عرضها في هذه الوحدة

مثال فيما يلي سوف نستعرض كيفية الحصول على شيفرة الالتفاف CONVOLU-TIONAL CODES والتي يتم توليدها لسلسلة من الخانات الثنائية وذلك باستخدام مرمز الالتفاف والذي يحتوي على مسجل بالاضافة الى جامعين من نوع (mod 2).

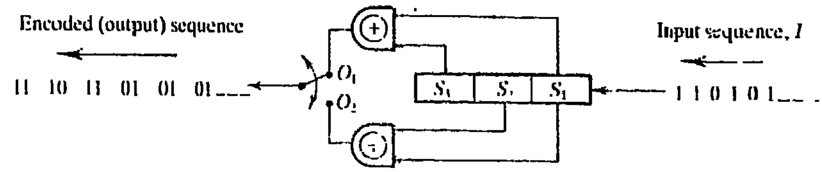
من خلال دائرة المرمز يتبين لنا ان كل خانة من خانات السلسلة المدخلة تستبدل بخانتين من شيفرة الالتفاف

$S1 \oplus S3$

$S1 \oplus S2$

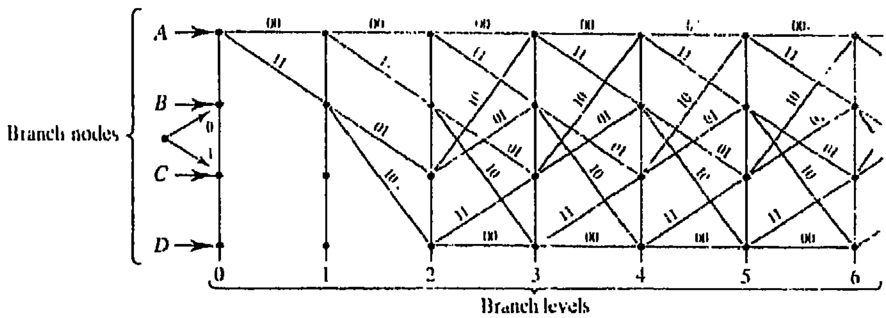
والشكل التالي يوضح دائرة المرمز وكيفية توليد شيفرة الالتفاف.

(a)

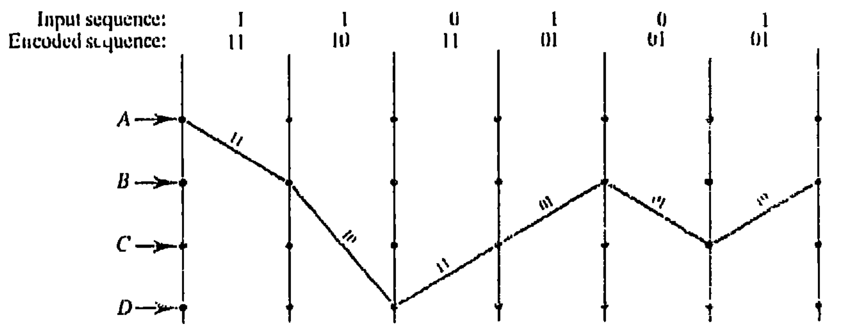


Input sequence I	Shift register contents S_1	S_2	S_3	Output sequence O_1	O_2
1	1	0	0	1	1
1	1	1	0	1	0
0	0	1	1	1	1
1	1	0	1	0	1
0	0	1	0	0	1
1	1	0	1	0	1
⋮		⋮		⋮	

(b)



(c)



والبرنامج التالي يوضح كيفية الحصول على شيفرة الالتفاف لسلسلة البيانات الثنائية المدخلة.

```
;****     STACK    SEGMENT
STACK     SEGMENT PARA    STACK    'STACK'
          DW      32 DUP(?)
STACK     ENDS
;****     DATA    SEGMENT           ****************************
DAT       SEGMENT
DATAIN    LABEL   BYTE
MAXLEN    DB      2 ;**** JUST ONE NUMBER ACCEPTED   *********
ACTLEN    DB      ?
X         DB      2 DUP(' '),'$';***  A LLOCATION FOR INPUT DATA   ***
DATSAV    DB      2 DUP(' '),13,10,'$';** SPACE FOR OUTPUT DATA**
CRLF      DB      13,10,'$';*** LINE FEED AND RETURN TO START OF LINE ***
MESG      DB      'ENTER  DATA :','$'
DAT       ENDS
;****     CODE    SEGMENT           ****************************
COD       SEGMENT
MAIN      PROC    FAR;*** PROCEDURE TO RETURN BACK TO (DOS) **********
;**********       INITIALIZATION  ***********************************
          ASSUME  DS:DAT,CS:COD,SS:STACK
          PUSH    DS
          SUB     AX,AX
          PUSH    AX
          MOV     AX,DAT
          MOV     DS,AX
          MOV     ES,AX
CALL      CLS;****         CLEAR    THE      SCREEN   ************
CALL      SET;****         SETTING THE       CURSOR   ************
CALL      ACCDAT;****      ACCEPTING         DATA     ************
          LEA     DI,DATSAV
          LEA     SI,X
          MOV     BX,3030H;***    SET ZERO BUT BY ASCII   ****
          MOV     DX,3030H
          MOV     AL,[SI];****    MOV DATA TO ACCUMULATOR ***
          MOV     AH,AL
          XOR     DH,AH  ;****  CONVOL. CODE OPERATION    ****
          OR      DH,30H ;****    CONVERT VALUE TO ASCII ****
          MOV     [DI],DH;****    SAVE    VALUE          ****
          XOR     BH,AH  ;****    SHIFT   BIT OF DATA    ****
          INC     DI
          OR      BH,30H ;****    CONVERT VALUE TO ASCII ****
          MOV     [DI],BH;****    SAVE    VALUE          ****
;****     STORE   REGISTERS       **************************
          PUSH    AX
          PUSH    BX
          PUSH    DX
CALL      OUTDAT          ;****   OUTPUT THE DATA ***********
;*********  RECLAIM      VALUE OF REGISTERS  ************************
          POP     DX
          POP     BX
          POP     AX
          MOV     BL,AL  ;**** SHIFT  DATA  *****************
          MOV     BH,BL  ;**** TO SAVE DATA *****************
          MOV     DX,3030H;*** MOV ZERO BUT BY ASCII ********
          MOV     CX,11   ;*** NUMBER OF DATA BITE 12 *******
L1:
;**********       STORE   REGISTERS  ********************************
          PUSH    AX
          PUSH    BX
          PUSH    DX
CALL      ACCDAT ;****  ACCEPTING DATA  ********************
;********  RECLAIM  VALUE  OF REGISTERS  **********************
          POP     DX
          POP     BX
          POP     AX
          MOV     AL,[SI];****  SAVE ACCEPTD DATA   **********
```

```
        XOR     BH,AH   ;****  CONV. CODE OPERATION ********
        OR      BH,30H  ;****  CONVERT VALUE TO ASCII ******
        INC     DI
        MOV     [DI],BH;****  SAVE OUT DATA  **************
;**********  STORE   REGISTERS  ********************************
        PUSH    AX
        PUSH    BX
        PUSH    DX
CALL    OUTDAT ;****  OUT DATA  ***************************
;**********  RECLAIM   REGISTERS  ******************************
        POP     DX
        POP     BX
        POP     AX
        MOV     DL,BL   ;****  SHIFT DATA  ****************
        MOV     BL,AL   ;****  SHIFT DATA  ****************
        MOV     BH,BL   ;****  SAVE  DATA  ****************
        MOV     DH,DL   ;****  SAVE  DATA  ****************
        LOOP    L1      ;****  JUMP WHILE CX<>0 ************
        RET
MAIN    ENDP            ;****  END MAIN PROGRAM ************
;****  PROCEDURE  CLEAR  SCREEN  ******************************
CLS     PROC
        MOV     AX,0600H
        MOV     BH,07
        MOV     CX,00
        MOV     DX,184FH
        INT     10H
        RET
CLS     ENDP
;****  PROCEDURE  SET  CURSOR  *********************************
SET     PROC
        MOV     AH,02
        MOV     BH,00
        MOV     DX,00
        INT     10H
        RET
SET     ENDP
;****  PROCEDURE  ACCEPT  DATA  ********************************
ACCDAT  PROC
        MOV     AH,09
        LEA     DX,MESG
        INT     21H
        MOV     AH,10
        LEA     DX,DATAIN
        INT     21H
        RET
ACCDAT  ENDP
;**********  OUTPUT  DATA  *************************************
OUTDAT PROC
        MOV     AH,09H
        LEA     DX,CRLF
        INT     21H
        MOV     AH,09
        LEA     DX,DATSAV
        INT     21H
        RET
OUTDAT  ENDP
        COD     ENDS
        END     MAIN

ENTER   DATA :1
11
ENTER   DATA :1
10
```

```
ENTER    DATA  :1
00
ENTER    DATA  :0
11
ENTER    DATA  :0
10
ENTER    DATA  :1
11
ENTER    DATA  :1
10
ENTER    DATA  :0
11
ENTER    DATA  :0
10
ENTER    DATA  :0
00
ENTER    DATA  :1
11
ENTER    DATA  :0
01
```

مثال : فيما يلي نستعرض برنامجا لتحويل مجموعة من الخلايا الثنائية الى شيفرة هامينج (Hamming code) حيث يمكن باستخدام هذه الشيفرة تحديد الخطأ في البيانات (المدخلة) المرسلة وتحديد موقع الخطأ.

فمثلا لتحويل الخانات الثنائية 1100010 الى شيفرة هامينج نتبع الآتي :

11	10	9	8	7	6	5	4	3	2	1
1	1	0	x	0	0	1	x	0	x	x

نجمع الخانات المناظرة للواحد مع اهمال الحمل (XOR)

$11 \oplus 10 \oplus 5$

1011

1010

0101

(4) 0100

وبهذا فان شيفرة هامينج تصبح كما يلي

11	10	9	8	7	6	5	4	3	2	1
1	1	0	0	0	0	1	1	0	0	0

ويمكن استخدام هذه الشيفرة لتحديد الخطأ وموقع هذا الخطأ . فلو افترضنا وجود 1 مقابل الخانة السادسة فان نتيجة جمع الخانات تصبح

1011

1010

0110

0101

0100

0110

وبما ان نتيجة الجمع لا تساوي صفر فان البيانات تحوي خطأ موقعه يُحدد بنتيجة الجمع (٦ في المثال)

```
dat segment
    X DW (?)
    Y DW (?)
    M DW (?)
    F DW (?)
    K DB (?)
    N DW (?)
    Z DW 0,1,2,4,8,10H,20H,40H,80H,100H,200H,400H
DAT ENDS
COD SEGMENT
    ASSUME DS:DAT,CS:COD
    PUSH DS
    SUB AX,AX
    PUSH AX
    MOV AX,DAT
    MOV DS,AX
    MOV AX,0100101B;DATA INPUT(7 bit)
    MOV X,Ax;memory location x contain input data to use it-
    SUB AX,AX;  when we need it.
    MOV BX,X
    MOV AX,X
    MOV DX,X
    MOV CL,4
    SHL BX,CL
    DEC CL
    SHL AX  ,CL
    DEC CL
    SHL DX,CL
    AND BX,0700H
    AND AX,0070H
    AND DX,0004H
    OR BX,AX
    OR BX,DX
    MOV AX,BX
    MOV Y,AX;location y contain the number which combon-
    MOV Cx,0;  11bit but bits number 1,2,4,8 is empty.
    MOV BX,0
    LEA DI,Z
L1:INC DI
    CMP CX,11
    JZ L3
    INC CX
L2:TEST AX,[DI]
    JNZ L1
    XOR BX,CX
    INC CX
    INC DI
    CMP CX,11
```

```
    JNZ L2
L3:MOV M,BX
    MOV AX,BX
    MOV CX,BX
    AND BX,3
    MOV DX,BX
    SHL AX,1
    SHL AX,1
    SHL AX,1
    SHL AX,1
    SHL CX,1
    AND AX,0080H
    AND CX,0008H
    OR AX,CX
    OR AX,DX
    MOV BX,AX
    MOV AX,Y
    OR AX,BX
    MOV F,AX;This is the end of cod ing to 11bit and this-
    LEA DI,Z; number will be transfer it to channel
L4:INC DX
    CMP CX,11
    JZ L7
    INC CX
L5:TEST AX,[DI]
    JNZ L4
    XOR BX,CX
    INC CX
    INC DI
    CMP CX,11
    JNZ L5
    CMP BX,0;In this operation we make sure that we havent-
    JZ L7;  any distortion to the data input.
   DEC BX
    MOV cx,bx
    MOV DX,1
    SHL DX,CL
    XOR AX,CX
L7:MOV BX,F
    MOV AX,F
    MOV DX,F
    MOV CL,4
    SHR BX,CL
    DEC CL
    SHR AX,CL
    DEC CL
    SHR DX,CL
    AND BX,0070H
    AND AX,14
    AND DX,0001
    OR BX,AX
    OR BX,DX
    MOV AX,BX; Register ax is contain the data input which-
    MOV N,AX
HLT;                     we input it.
COD ENDS
END
```

```
AX=0025  BX=0025  CX=0002  DX=0001  SP=FFFC  BP=0000  SI=0000  DI=000D
DS=1317  ES=1307  SS=1317  CS=1317  IP=0101   NV UP DI PL NZ NA PO NC
1317:0101 A30900        MOV     [0009],AX                    DS:0009=0000

AX=0025  BX=0025  CX=0002  DX=0001  SP=FFFC  BP=0000  SI=0000  DI=000D
DS=1317  ES=1307  SS=1317  CS=1317  IP=0104   NV UP DI PL NZ NA PO NC
1317:0104 F4            HLT

AX=0025  BX=0025  CX=0002  DX=0001  SP=FFFC  BP=0000  SI=0000  DI=000D
DS=1317  ES=1307  SS=1317  CS=1317  IP=0105   NV UP DI PL NZ NA PO NC
1317:0105 0A24          OR      AH,[SI]                      DS:0000=25

^C

-D DS:0
1317:0000  25 00 24 02 0E 00 AE 02-00 25 00 00 00 01 00 02   %.$......%......
1317:0010  00 04 00 08 00 10 00 20-00 40 00 80 00 00 01 00   ....... .@......
1317:0020  02 00 04 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
1317:0030  1E 2B C0 50 B8 17 13 8E-D8 B8 25 00 A3 00 00 2B   .+@P8...X8%.#..+
1317:0040  C0 8B 1E 00 00 A1 00 00-8B 16 00 00 B1 04 D3 E3   @....!......1.Sc
1317:0050  FE C9 D3 E0 FE C9 D3 E2-81 E3 00 07 25 70 00 83   ~IS'~ISb.c..%p..
1317:0060  E2 04 0B D8 0B DA 8B C3-A3 02 00 B9 00 00 BB 00   b..X.Z.C#..9..;.
1317:0070  00 8D 3E 0B 00 47 83 F9-0B 74 0E 41 85 05 75 F5   ..>..G.y.t.A..uu
-
```

الوحدة السابعة

معالجة السلاسل

- التحريك
- التحميل
- التخزين
- المقارنة
- الاستعراض

معالجة السلاسل
STRING PROCESSING

تعرف السلسلة على انها مجموعة من الرموز قد تتضمن ارقاما او احرفا او خليطا من الارقام والاحرف والرموز الخاصة وقد يتطلب الامر في بعض الاحيان معالجة سلاسل بأطوال مختلفة وقد تزيد عن الكلمة وفي هذه الحالة تستخدم تعليمات خاصة لمعالجة السلاسل منها :

- نقل محتوى بايت او كلمة من موقع في الذاكرة الى موقع آخر (MOVSB)
- تحميل AL أو AX ببايت او كلمة من موقع في الذاكرة (LODS)
- تخزين محتوى AL أو AX في موقع في الذاكرة (STOS)
- مقارنة محتوى موقعين في الذاكرة (CMPS)
- مقارنة محتوى AL أو AX بمحتوى موقع في الذاكرة (SCAS)

التعليمة MOVSB (التحريك)

تستخدم هذه التعليمة لنقل محتوى بايت من موقع الى موقع اخر في الذاكرة عادة ما تقترن هذه التعليمة بتعليمة التكرار REP وعند استخدام هذه التعليمة لا بد من مراعاة ما يلي :

- تحديد بداية النقل (من اليسار أو اليمين) وذلك بالتأثير على بت الاتجاه (DF) حيث يمكن استخدام الامر CLD (clear DF) وذلك لاجراء عملية النقل من اليسار او استخدام الامر (STD) (set DF) لاجراء عملية النقل من اليمين .
- تحديد عدد البايت في الحقل المرسل ونسخ هذه القيمة في CX .
- الحقل المرسل يعنون باستخدام SI
- الحقل المستقبل يعنون باستخدام DI

-استخدام الامر REP MOVSB لنقل محتويات الحقل المرسل الى المستقبل وذلك بتكرار عملية نقل البايت وطرح واحد اوتوماتيكيا بعد كل عملية نقل من CX الى ان تصبح قيمته مساوية للصفر

ويمكن تمثيل هذا بالتعليمات التالية

```
SENDFLD
RECFLD    DB   10   DUP ('M')
          DB   10   DUP ('  ')
          .
          .

          CLD
          MOV  CX , 10
          LEA DI , RECFLD
          LEA   SI , SENDFLD
          REP MOVSB
```

وتستخدم MOVSB لتسهيل عمليات النقل حيث يمكن ان تحل التعليمة MOVSB REP محل التعليمات التالية

```
LOOP2:    JCXZ   LOOP1
          LOOP2 : MOV AL , [SI]
          MOV  [DI], A
          INC   SI
          INC   DI
          LOOP    LOOP2
Loop: ...
```

اما التعليمة MOVSW فتشبه التعليمة MOVSB لكن يُفضل استخدامها عندما يكون عدد البايت في الحقل المرسل زوجيا اما اذا كان فرديا فيفضل استخدام MOVSB

التعليمة LODS (التحميل)

تستخدم هذه التعليمة لتحميل AL أو AX ببايت او كلمة من موقع في الذاكرة مشار اليه بالمسجل SI وتأخذ هذه التعليمة شكلين :

١ – LODSB لتحميل بايت في المسجل AL .

٢ – LODSW لتحميل كلمة في المسجل AX .

التعليمة STOS (التخزين)

تشبه هذه التعليمة LODS الا انها تستخدم لتحميل موقع في الذاكرة من المسجل AL أو AX اعتمادا على طول الموقع (بايت او كلمة) وتأخذ هذه التعليمة شكلين :

١ – LODSB لتحميل محتوى AL في موقع في الذاكرة .

٢ – LODSW لتحميل محتوى AX في موقع في الذاكرة .

التعليمة CMPS (المقارنة)

تستخدم هذه التعليمة لمقارنة محتوى موقع معنون بالمسجل SI مع محتوى موقع آخر معنون بالمسجل DI حيث تتم مقارنة بايت بايت من كل من SI أو DI عند استخدامCMPSB او كلمة كلمة عند استخدام CMPSW وترتبط هذه التعليمة بأمر التكرار REP وفي الغالب ما يستخدم الامر REPE لتكرار عملية المقارنة ما دام شرط المساواة قائما بحيث يتم قطع عملية المقارنة في حالة عدم المساواة .

التعليمةSCAS (الاستعراض)

تختلف هذه التعليمة عن تعليمة cmps في انها تقوم بالتأكد من امكانية احتواء سلسلة على بايت (حرف) او كلمة مخزنة في AL أو AX وعادة ما ترتبط هذه التعليمة مع الامر REPNE لاجراء عمليات البحث في حالة عدم المساواة .

وقد تستخدم هذه التعليمة عند اجراء عملية استبدال بايت معين حيث يتم البحث عن هذا البايت اعتمادا على المحتوى المطلوب وعند ايجاده يتم استبداله بمحتوى جديد كما يلي

```
STRLN     EQU  10
  STR1    DB   "MY NAME * IS'
          .
          .
         CLD
         MOV   AL, '*'
         MOV   CX, STRLEN
LEA   DI , STR1
         REPNE   SCASB
         JNZ  EX1
         DEC  DI
         MOV   BYTE PTR[DI] , 20H
EX1 :   RET
```

حيث يتم البحث عن * واستبدالها بالفراغ في حالة وجودها .
والبرامج التالية توضح اهم العمليات التي يمكن اجراؤها على السلاسل

```
; ILLUSTRATION OF STRING OPERATIONS
;------------------------------------
STACKSG SEGMENT PARA STACK 'STACK'
                DW      32 DUP(' ')
STACKSG ENDS
;-------------------------------------
DATASG  SEGMENT PARA 'DATA'
NAME1   DB      'ZIADALQADI'
NAME2   DB      10 DUP(' ')
NAME3   DB      10 DUP(' ')
DATASG  ENDS
;--------------------------------------
CODESG  SEGMENT PARA 'CODE'
BEGIN   PROC    FAR
                ASSUME CS:CODESG,DS:DATASG,SS:STACKSG,ES:DATASG
                PUSH    DS
                SUB     AX.AX
                PUSH    AX
                MOV     AX,DATASG
                MOV     DS,AX
                MOV     ES,AX
                CALL    PMOVSB
                CALL    PMOVSW
                CALL    PLODS
                CALL    PSTCS
                CALL    PCMPS
                CALL    PSCAS
                RET
BEGIN   ENDP
;       USE OF MOVSB
;       ------------
PMOVSB  PROC    NEAR
                CLD
                LEA     SI,NAME1
                LEA     DI,NAME2
                MOV     CX,10
                REP MOVSB
                RET
PMOVSB  ENDP
;        USE OF MOVSW INSTRUCTION
;        ------------------------
PMOVSW  PROC    NEAR
                CLD
                LEA     SI,NAME2
                LEA     DI,NAME3
                MOV     CX,05
                REP MOVSW
                RET
```

```
PMOVSW   ENDP
;            USE OF LOAD(LODSW) INSTRUCTION
;        ------------------------------
PLODS    PROC      NEAR
                   CLD
                   LEA      SI,NAME1
                   LODSW
                   RET
PLODS          ENDP
;        USE OF STORE (STOSW) INSTRUCTION
;        ----------------------------  --
PSTOS    PROC      NEAR
                   CLD
                   LEA      DI,NAME3
                   MOV      CX,05
                   MOV      AX,2020H
                   REP STOSW
                   RET
PSTOS    ENDP
;          USE OF COMPARE (CMPS) INSTRUCTION
;          ---------------------------------
PCMPS    PROC      NEAR
                   CLD
                   MOV      CX,10
                   LEA      SI,NAME1
                   LEA      DI,NAME2
                   REPE CMPSB
                   JNE      Z20
                   MOV      BH,01
Z20:
                   MOV      CX,10
                   LEA      SI,NAME2
                   LEA      DI,NAME3
                   REPE CMPSB
                   JE       Z30
                   MOV      BL,02
Z30:
                   RET
PCMPS          ENDP
;        USE OF SCAN (SCASB) INSTRUCTOIN
;        ------------------------------
PSCAS    PROC      NEAR
                   CLD
                   MOV      CX,10
                   LEA      DI,NAME1
                   MOV      AL,'A'
                   REPNE SCASB
                   JNE      ZL10
                   MOV      AH,03
ZL10:
                   RET
PSCAS    ENDP
CODESG   ENDS
                   END      BEGIN
```

```
;     RIGHT-ADJUST DISPLAYED NAMES
;--------------------------------
CODESG   SEGMENT PARA 'CODE'
                 ASSUME CS:CODESG,DS:CODESG,SS:CODESG,ES:CODESG
                 ORG      100H
BEGIN:   JMP     SHORT MAIN
;------------------------------------
NPAR             LABEL    BYTE
MAXL             DB       31
ACCN             DB       ?
NMFL             DB       31 DUP(' ')
MESS             DB       'NAME?', '$'
NMDS             DB       31 DUP(' '), 13, 10, '$'
ROW              DB       00
;-------------------------------------
MAIN             PROC     NEAR
                 MOV      AX,0600H
                 CALL     PSCR
                 SUB      DX,DX
                 CALL     PCURS
Z10LOOP:
                 CALL     PINPT
                 TEST     ACCN,0FFH
                 JZ       Z90
                 CALL     PSCAS
                 CMP      AL,'*'
                 JE       Z10LOOP
                 CALL     PRGHT
                 CALL     PCLNM
                 JMP      Z10LOOP
Z90:
                 RET
MAIN             ENDP
;     MESSAGE FOR INPUT NAME
;     ----------------------
PINPT    PROC
                 MOV      AH,09
                 LEA      DX,MESS
                 INT      21H
                 MOV      AH,0AH
                 LEA      DX,NPAR
                 INT      21H
                 RET
PINPT    ENDP
;       SCAN NAME FOR ASTERISK
;       ----------------------
PSCAS    PROC
                 CLD
                 MOV      AL,'*'
                 MOV      CX,30
                 LEA      DI,NMFL
                 REPNE SCASB
                 JE       D20
                 MOV      AL,20H
D20:             RET
```

```
PSCAS    ENDP
;    RIGHT ADJUST AND DISPLAY NAME
;    ---------------------------
PRGHT    PROC
                  STD
                  SUB      CH,CH
                  MOV      CL,ACCN
                  LEA      SI,NMFL
                  ADD      SI,CX
                  DEC      SI
                  LEA      DI,NMDS+30
                  REP      MOVSB
                  MOV      DH,ROW
                  MOV      DL,48
                  CALL     PCURS
                  MOV      AH,09
                  LEA      DX,NMDS
                  INT      21H
                  CMP      ROW,20
                  JAE      E20
                  INC      ROW
                  JMP      E90
E20:
                  MOV      AX,0601H
                  CALL     PSCR
                  MOV      DH,ROW
                  MOV      DL,00
                  CALL     PCURS
E90:              RET
PRGHT    ENDP
;          CLEAR NAME
;          ----------
PCLNM    PROC
                  CLD
                  MOV      AX,2020H
                  MOV      CX,15
                  LEA      DI,NMDS
                  REP      STOSW
                  RET
PCLNM    ENDP
;          SCROLL SCREEN
;          -------------
PSCR              PROC
                  MOV      BH,30
                  MOV      CX,00
                  MOV      DX,184FH
                  INT      10H
                  RET
PSCR              ENDP
;          SET CURSOR/COL
;          --------------
PCURS    PROC
                  MOV      AH,02
                  SUB      BH,BH
                  INT      10H
                  RET
PCURS    ENDP
CODESG   ENDS
                  END      BEGIN
```

الوحدة الثامنة

الجداول

- البحث
- البحث باستخدام السلاسل
- الترجمة (XLAT)
- فرز عناصر الجدول

الجداول
TABLES

يُعرف الجدول على انه مجموعة من الداخل على ان يحتوي كل مدخل منها عنصرا او اكثر من عناصر البيانات مع مراعاة تشابه العناصر في المداخل كافة في النوع .

يتم الاعلان عن الجدول وذلك باعطائه اسما ومن ثم يتم تحديد عناصر الجدول باستخدام التعليمات الخاصة لتعريف البيانات والامثلة التالية توضح مفهوم الجداول.

```
DAYTAB   DB   'SAT' , 'SUN' , ... , 'FRI'
TAB1   DB   200, 210, 120, ...
```

فالجدول الاول يحتوي على مجموعة من العناصر الابجدية بحيث يتم تخصيص ٣ بايت للعنصر الواحد اما الجدول الثاني فيضم مجموعة من الثوابت حيث خصص بايت واحد لكل عنصر .

كما ويمكن ان يحتوي كل مدخل من مداخل الجدول على خليط من العناصر كما يلي

```
TABM   DB 1 , 'COBOL'
       DB 2 , 'BASIC'
       DB 3 , 'PASCAL'
```

جيث يضم كل مدخل عنصرين الاول رقمي بطول بايت واحد والثاني ابجدي وبطول ٦ بايت (على اعتبار ان كل حرف يخصص له بايت واحد) .

نلاحظ من ما سبق ان الجدول يتضمن عنصر او مجموعة من العناصر وانه تم تحديد قيم المدخلات في الجدول وفي بعض الاحيان قد يتطلب الامر تحضير الجدول لاستخدامه لاغراض تخزين قيم يتم حسابها اثناء تنفيذ البرنامج ويتم الاعلان عن هذا الجدول في مقطع البيانات كما يلي :

```
TABN  DB  30  DUP (20 DUP (' '))
```

من خلال هذا المثال تتبين لنا الامور التالية :

- اسم الجدول TABN
- يحتوي هذا الجدول على ٣٠ مدخلا (عنصرا) .
- طول كل مدخله ٢٠ بايتا .

وتمتاز الجداول بخاصية المعالجة المباشرة (Direct access) حيث يمكن استرجاع عنصر ما من عناصر الجدول وذلك بتحديد موقعه فمثلا لو أخذنا الجدول التالي :

```
CTAB DB 'ASSEMBLER'
     DB 'COBOL'
     DB 'BASIC'
     .
     .
```

فإن العنصر الاول يقع في الموقع (CTAB + 0) اما العنصر الثاني (COBOL) فيقع في الموقع (CTAB + 9) وهكذا ، ويمكن الرجوع الى عنصر من عناصر هذا الجدول وذلك بذكر رقمه مع مراعاة تحويل هذا الرقم من نظام (ASCII) الى النظام الثنائي عند ادخال هذا الرقم عن طريق لوحة المفاتيح . فلو اردنا مثلا استرجاع العنصر الثالث فإن عملية تحديد موقعه تتم كما يلي

- حول الرقم من (ASCII) الى ثنائي(اي من صيغة ٣٣٣٠ الى 03).
- اطرح واحد من الرقم (2 = 1 - 3)
- اضرب الناتج في طول العنصر (18 = 9 . 2)
- اضف ناتج الضرب الى عنوان CTAB للحصول على عنوان العنصر المطلوب (CTAB + 18) .

وعملية المعالجة المباشرة هذه لا تعني استعراض الجدول والبحث عن العنصر المطلوب بل تعني حساب موقع هذا العنصر واسترجاع القيمة المخزنة في هذا الموقع . والمثال التالي يستخدم جدولا مؤلفا من ١٢ عنصرا تمثل اسماء الاشهر وبطول ٣ بايت لكل عنصر وقد استخدم الرقم "١١" وذلك لاسترجاع العنصر الحادي عشر حيث تضمن البرنامج عملية تحويل هذا الرقم من ASCII الى الثنائي وبعدها تم حساب موقع العنصر الحادي عشر ثم عرضت القيمة المخزنة في هذا الموقع على الشاشة .

```
                PAGE 30,60
TITLE   DIRECT TABLE ACCESS
;DIRECT ACCESS OF A TABLE OF THE MONTHS NAME
;========================================
CODESG  SEGMENT PARA 'CODE'
                ASSUME CS:CODESG,DS:CODESG,SS:CODESG,ES:CODESG
                ORG     100H
BEGIN:  JMP     SHORT MAIN
;       TABLE DEFFINITION
;       =================
THREE   DB      3
MONIN   DB      '11'
ALFMON  DB      '???','$'
MONTAB  DB      'JAN','FEB','MAR','APR','MAY','JUN'
                DB      'JUL','AUG','SEP','OCT','NOV','DEC'
;==========================================================
MAIN            PROC    NEAR
                CALL    CONV1       ;CONVERT TO BINARY
                CALL    LOC1        ;LOCATE MONTH
                CALL    DISP1       ;DISPLAY ALPHA MONTH
MAIN            ENDP
;           CONVERT ASCII TO BINARY
;           =====================
CONV1   PROC
                MOV     AH,MONIN     ;SET UP MONTH
                MOV     AL,MONIN+1
                XOR     AX,3030H      ;CLEAR ASCII
                CMP     AH,00         ;MONTH 01-09 ?
                JZ      C20
                SUB     AH,AH
                ADD     AL,10
C20:            RET
CONV1   ENDP
;           LOCATE MONTH IN TABLE
;           ====================
LOC1            PROC
                LEA     SI,MONTAB
                DEC     AL              ;CORRECT FOR TABLE
                MUL     THREE           ;MULTIPLY AL BY 3
                ADD     SI,AX
                MOV     CX,03           ;INITIALIZE 3 CHAR
                CLD
                LEA     DI,ALFMON
                REP     MOVSB
                RET
LOC1            ENDP
;           DISPLAY ALPHA MONTH
;           ==================
DISP1   PROC
                LEA     DX,ALFMON
                MOV     AH,09
                INT     21H
                RET
DISP1   ENDP
CODESG  ENDS
                END     BEGIN
```

البحث في الجداول
TABLE SEARCHING

تختلف عملية البحث عن عملية المعالجة المباشرة في ان عملية البحث تتضمن البحث عن قيمة عنصر معين وذلك باجراء المقارنات اللازمة ثم استخراج العناصر المرتبطة بالعنصر المطلوب في حين تتضمن المعالجة المباشرة عملية حساب موقع العنصر المطلوب وذلك من خلال معرفة ترتيب هذا العنصر في الجدول . فلو اخذنا الجدولين التاليين

```
ITEMNO   DB '101', '108', '115', ...
ITEMNAME DB 'AAAA', 'BBBB', 'CCCC'
```

واردنا استرجاع معلومات عن العنصر رقم ١٠٨ فان هذا الرقم سوف يقارن بالعنصر الاول وكونه لا يساويه يقارن بالعنصر الثاني وكون العنصر المطلوب يساوي هذا العنصر فانه يتم استرجاع العنصر المرتبط معه والواقع في الموقع الثاني (BBBB).

لاحظ ان الجدولين السابقان يمكن كتابتهما بالصورة التالية

```
ITEMNO  DB '101' , 'AAAA'
        DB  '108' , 'BBBB'
        DB '115' , 'CCCC'
```

والبرنامج التالي يبين كيفية استرجاع المادة التي تحمل الرقم ٢٣ ومن خلال تتبع هذا البرنامج يمكن ملاحظة الامور التالية

– الرقم 23 يتم تخزينه بالشكل 3233 وعند تحميل هذا الرقم في AX فانه سوف يخزن بشكل معكوس (3332) اي ان 32 سوف تخزن في AL و 33 في AH ولهذا السبب تم استخدام امر التغير XCHG وذلك لارجاع الرقم الى 3233 .

– عدد المقارنات الاكبر هو ٦ نظرا لان عدد العناصر يساوي ٦

– في حالة عدم توفر الرقم المطلوب يتم الخروج بعد تنفيذ روتين معين يبين عدم توفر العنصر في الجدول .

– عملية الانتقال الى المقارنة الثانية تتم بإضافة الرقم ١٢ الى SI نظرا لان طول العناصر في المدخل الواحد يساوي ١٢ (٢ + ١٠)

– اسم المادة يقع في موقع يبعد عن رقمها بالمقدار ٢ فمثلا اسم المادة الاولى يقع في الموقع STOCFTAB + 2 ولهذا السبب استخدم الامران المتكرران

```
INC SI
INC SI
```

```
                PAGE    30,60
TITLE   TABLE SEARCHING
CODESG  SEGMENT PARA 'CODE'
                ASSUME  CS:CODESG,DS:CODESG,SS:CODESG,ES:CODESG
                ORG     100H
BEGIN:  JMP     SHORT MAIN
;           T A B L E
            ==========
;
NAMEIN  DW      '23'
NAMETAB DB      '05','ZIAD'
                DB      '08','FATTAH'
                DB      '09','SAMI'
                DB      '12','IBRAHEEM'
                DB      '23','RASHAD'
                DB      '27','NATASHA'
DSNAME  DB      10 DUP(?)
;====================================
MAIN            PROC    NEAR
                MOV     AX,NAMEIN    ;GET NUMBER
                XCHG    AL,AH
                MOV     CX,6         ;NUMBER OF NAMES
                LEA     SI,NAMETAB   ;INITIALIZE TABLE ADDRESS
A20:
                CMP     AX,[SI]     ; NUMBER : TABLE
                JE      A30         ;EQUAL  EXIT
                ADD     SI,12
                LOOP    A20
                CALL    ERR1
                RET
A30:
                MOV     CX,05
                LEA     DI,DSNAME
                INC     SI
                INC     SI
                REP     MOVSW
MAIN            ENDP
ERR1            PROC
;           DISPLAY ERROR MESSAGE
                RET
ERR1            ENDP
CODESG  ENDS
                END     BEGIN
```

ويمكن التعامل مع عناصر الجدول اعتمادا على مدى (Range) معين فمثلا الجدول التالي يبين مجموع المبيعات بالدينار والعلاوة المستحقة والتي تعتمد على مجموع المبيعات

مجموع المبيعات	العلاوة
٠ – ١٠٠٠	١٠
١٠٠١ – ٢٠٠٠	٢٥
٢٠٠١ – ٥٠٠٠	٤٠
٥٠٠١ – ٩٩٩٩	٦٠

حيث يمكن تمثيل هذه الجداول كما يلي

TSALTAB DW 1000, 2000 , 5000 , 9999

ALTAB DB 10, 25 , 40 , 60

فلحساب العلاوة المستحقة يؤخذ مجموع المبيعات ويقارن بأول عنصر في جدول المبيعات فاذا كان المجموع اقل او يساوي من قيمة المجموع في الجدول تؤخذ القيمة المناظرة للعلاوة في جدول العلاوات وإلا تتم عملية المقارنة مع العنصر الثاني وهكذا .

البحث باستخدام السلاسل

في بعض الاحيان قد يزيد طول عنصر البيانات في الجدول عن ٢ بايت وفي هذه الحالة يمكن اعتبار هذا العنصر سلسلة وللوصول الى مدخلة معينة في الجدول يتم تحديد الرقم المطلوب حيث يتم تخزينه في المسجل DI اما اول عنصر في الجدول فيتم تخزينه في الموقع المشار اليه بالمسجل SI وبعدها تتم عملية مقارنة ما هو مخزن في الموقع DI و SI بايت بايت باستخدام التعليمة الخاصة بالسلسلة REPE COMS فاذا اختلف بايت فيهما يتم الانتقال الى العنصر التالي في الجدول وهكذا فلو اخذنا الجدول التالي

```
TABI  DB '100' , 'AAAA' , '200' , 'BBBB'
      DB '300' , 'CCCC', '400' , 'DDDD'
```

واردنا البحث عن العنصر ٣٠٠ فان العملية سوف تتم كما يلي

- يخزن عنوان العنصر ٣٠٠ في المسجل DI
- يخزن عنوان بداية الجدول في المسجل SI
- يخزن العدد ٣ (طول المقارنة) في المسجل CX
- تتم عملية مقارنة DI , SI باستخدام REPE COMS
- نتيجة عملية المقارنة (مقارنة البايت الاول) غير متطابقة مما يعني عدم تساوي الرقمين لذا ننتقل الى العنصر الثاني في الجدول وذلك باضافة محتوى CX الى SI وطول الحقل الابجدي الىSI (٢ + ٤ = ٦) للوصول الى العنصر التالي ويتم تصفير DI ووضع ٣ فيCX لتكرار عملية المقارنة التالية وهكذا .
- عند الوصول الى الرقم المطلوب يتم نقل الحقل الابجدي كسلسلة باستخدام اوامر نقل السلاسل (MOVSW مثلا) .

- عند عدم توفر العنصر المطلوب يتم الخروج وذلك بالاشارة الى عدم توفر الرقم المطلوب .

والمثال التالي يوضح استخدام هذه المفاهيم لاسترجاع اسم الطالب وذلك باستخدام رقمه والمساوي لـ ١٥٠ .

```
; TABLE SEARCHING USING STRINGS
;==================================
CODESG   SEGMENT PARA 'CODE'
                 ASSUME CS:CODESG,DS:CODESG,SS:CODESG,ES:CODESG
                 ORG     100H
BEGIN:   JMP     SHORT MAIN
;------- DATA TABLE-----------------
STNO             DB      '150'
STTAB    DB      '050','MUHAMED ';START OF TABLE
                 DB      '100','ALI     '
                 DB      '150','ZIAD    '
                 DB      '180','NATASHA '
                 DB      '183','FATTAH  '
                 DB      '190','LENA    '
                 DB      '999',8 DUP(' ')     ;END OF TABLE
NOUT             DB      8 DUP(?)
;=====================================
MAIN             PROC NEAR
                 CLD
                 LEA     SI,STTAB
Z20:
                 MOV     CX,03
                 LEA     DI,STNO
                 REPE    CMPSB
                 JE      Z30
                 JA      Z40
                 ADD     SI,CX
                 ADD     SI,08
                 JMP     Z20
Z30:
                 MOV     CX,04
                 LEA     DI,NOUT
                 REP     MOVSW
                 RET
Z40:
                 CALL    ZERR
                 RET
MAIN             ENDP
ZERR             PROC
;     DISPLAY ERROR MESSAGE
                 RET
ZERR             ENDP
CODESG   ENDS
                 END     BEGIN
```

التعليمة XLAT

تستخدم هذه التعليمة لترجمة محتوى بايت من صيغة لأخرى فمثلا يمكن استخدام هذه التعليمة للترجمة من ASCII الى EBCDIC حيث يستخدم جدولا يتضمن الرموز ، بصيغة EBCDIC ويتم بواسطة هذا الجدول تحويل البيانات من ASCII الى EBCDIC وتتم عملية الترجمة هذه كما يلي :

- يتم استخدام BX لتحميل عنوان الجدول .
- البايت المراد تحويله من ASCII يحمل في AL
- تستخدم التعليمة محتوى AL لتحديد العنصر المناظر من الجدول وذلك باضافة محتوى AL الى BX ويمكن التعبير عن هذه الامور بالتعليمات التالية

```
LEA  BX , EBCDICTAB
MOV  AL , ASCIINO
XLAT
```

والبرنامج التالي يوضح كيفية استخدام هذه التعليمة لتحويل الرقم 31.5- من ASCII الى EBCDIC

```
; PROGRAM TO TRANSLATE ASCII TO EBCDIC
;------------------------------------------
CODESG   SEGMENT PARA 'CODE'
                 ASSUME CS:CODESG,DS:CODESG,SS:CODESG,ES:CODESG
                 ORG     100H
BEGIN:   JMP     MAIN
;------------------------------------------
ASCIINO DB       '-31.5 '
EBCDICNO         DB      6 DUP(' ')
TRANSTAB         DB      45 DUP(40H)      ;BLANKS
                 DB      60H,2DH
                 DB      5CH
                 DB      0F0H,0F1H,0F2H,0F3H,0F4H,0F5H,0F6H,0F7H
                 DB      0F8H,0F9H
                 DB      199 DUP(40H)
;------------------------------------------
MAIN             PROC    NEAR
                 LEA     SI,ASCIINO
                 LEA     DI,EBCDICNO
                 MOV     CX,06
                 LEA     BX,TRANSTAB
Z20:
                 MOV     AL,[SI]
                 XLAT
                 MOV     [DI],AL
                 INC     SI
                 INC     DI
                 LOOP    Z20
                 RET
MAIN             ENDP
CODESG   ENDS
                 END     BEGIN
```

فرز عناصر الجدول

TABLE SORTING

يُقصد بفرز عناصر الجدول ترتيبها ترتيبا تصاعديا او تنازليا اعتمادا على قيم عنصر من عناصر الجدول (المفتاح) .

تتضمن عملية الفرز (التصاعدي) مقارنة العنصر بالعنصر الذي يليه فاذا كانت قيمته اقل يتم الانتقال الى المقارنة التالية والا تتم عملية استبدال المواقع مع الاشارة الى حدوث عملية المبادلة وذلك بوضع ١ في مؤشر المبادلة وتتكرر عمليات المقارنة وفحص المؤشر حتى تصبح قيمته صفر ولا تتغير عند اجراء عمليات المقارنة (مما يعني ان العناصر قد رُتبت) .

والبرنامج التالي يوضح كيفية اجراء عملية الفرز حيث يتضمن هذا البرنامج :

- ادخال مجموعة من الاسماء عن طريق لوحة المفاتيح وتخزينها في جدول .
- ترتيب الاسماء في الجدول ترتيبا تصاعديا .
- عرض محتويات الجدول على الشاشة بعد اجراء عملية الفرز .

```
                PAGE 30,60
 ; PROGRAM TO SORT NAMES
 ;=====================
 STACK    SEGMENT PARA 'STACK'
                  DW      32 DUP(?)
 STACK    ENDS
 ;========================================
 DATASG   SEGMENT PARA 'DATA'
 NPAR             LABEL    BYTE     ;NAME PARAMETER LIST:
 MNLEN    DB  21           ;MAX LENGTH
 NLEN             DB  ?             ;NO OF CHATACTERS ENTERED
 NFLD             DB  21 DUP(' ') ;NAME
 CRLF             DB  13, 10, '$'
 ENDADDR DW   ?
 MESSG1   DB   'NAME ?','$'
 NCTR             DB   00
 NTAB             DB   30 DUP(20 DUP(' '))  ;NAME TABL
 NSAV             DB   20 DUP(?), 13, 10, '$'
 SWAPPED DB   00
 DATASG   ENDS
 ;========================================
 CODESG   SEGMENT PARA 'CODE'
 BEGIN    PROC    FAR
                  ASSUME CS:CODESG,DS:DATASG,SS:STACK,ES:DATASG
                  PUSH DS
                  SUB  AX,AX
                  PUSH AX
                  MOV AX,DATASG
                  MOV DS,AX
                  MOV ES,AX
```

```
                CLD
                LEA DI,NTAB
                CALL  ZCLR
                CALL   ZCURS
ZLOOP:
                CALL ZREAD
                CMP  NLEN,00
                JZ Z30
                CMP NCTR,30   ;30 NAMES ENTERED?
                JE Z30
                CALL ZSTOR
                JMP ZLOOP
Z30:
                CALL ZCLR
                CALL ZCURS
                CMP NCTR,01    ;ONE OR NO NAME ENTERED?
                JBE Z40
                CALL ZSORT
                CALL ZDISP
Z40:
             RET
BEGIN   ENDP
;         ACCEPT NAMES
;        ---------------
ZREAD   PROC
                MOV AH,09
                LEA DX,MESSG1
                INT 21H
                MOV AH,0AH
                LEA DX,NPAR
                INT 21H
                MOV AH,09
                LEA DX,CRLF      ;RETURN/LINE FEED
             INT 21H
                MOV BH,00
                MOV BL,NLEN      ;CLEAR CHAR AFTER NAME AND GET COUNT
                MOV CX,21
                SUB CX,BX       ;CALCULATE REMAINING LENGTH
Z20:
                MOV NFLD[BX],20H  ;SET TO BLANK
                INC BX
                LOOP Z20
                RET
ZREAD   ENDP
;
;            STORE NAMES IN TABLE
;            --------------------
ZSTOR   PROC
                INC NCTR    ;ADD TO NUMBER OF NAMES
                CLD
                LEA SI,NFLD
                MOV CX,10
                REP MOVSW      ;MOVE NAME TO TABLE
                RET
```

```
ZSTOR   ENDP
;
;               SORT NAMES IN TABLE
;               -------------------
ZSORT   PROC
                  SUB DI,40     ;SET UP STOP ADDRESS
                  MOV ENDADDR,DI
ZZ20:
                  MOV SWAPPED,00
                  LEA SI,NTAB
ZZ30:
              MOV CX,20          ;LENGTH OF COMPARE
                  MOV DI,SI
                  ADD DI,20          ;NEXT NAME FOR COMPARING
                  MOV AX,DI
                  MOV BX,SI
                  REPE CMPSB
                  JBE ZZ40
                  CALL ZXCHG         ;PERFORM EXCHANGING
ZZ40:
                  MOV SI,AX
                  CMP SI,ENDADDR     ;END OF TABLE?
                  JBE ZZ30
                  CMP SWAPPED,00      ;ANY SWAPS?
                  JNZ ZZ20
                  RET
ZSORT   ENDP
;           EXCHANGE TABLE ENTRIES
;           ----------------------
ZXCHG   PROC
                  MOV CX,10
                  LEA DI,NSAV
                  MOV SI,BX
                  REP MOVSW             ;MOVE LOWER ITEM TO SAVE
                  MOV CX,10
                  MOV DI,BX
                  REP MOVSW
                  MOV CX,10
                  LEA SI,NSAV
                  REP MOVSW
                  MOV SWAPPED,01
                  RET
ZXCHG   ENDP
;        DISPLAY SORTED NAMES
;        --------------------
ZDISP   PROC
                  LEA SI,NTAB
ZK20:
                  LEA DI,NSAV
                  MOV CX,10
                  REP MOVSW
                  MOV AH,09
                  LEA DX,NSAV
                  INT 21H
                  DEC NCTR
                  JNZ ZK20
                  RET
```

```
ZDISP    ENDP
;         CLEAR SCREEN
;         -------------
ZCLR               PROC
                   MOV AX,0600H
                   MOV BH,61H      ;COLOR 07 OF BW
                   SUB CX,CX
                   MOV DX,184FH
                   INT 10H
                   RET
ZCLR               ENDP
;           SET CURSOR
;           ----------
ZCURS    PROC
                   MOV AH,02
                   SUB BH,BH
                   SUB DX,DX
                   INT 10H
                   RET
ZCURS    ENDP
CODESG   ENDS
                   END      BEGIN
```

الوحدة التاسعة

تعليمات التحكم بالمعالج الدقيق

تعليمات التحكم بالمعالج الدقيق

يحتوي طاقم تعليمات لغة التجميع على تعليمات تسمح للمستخدم او المبرمج التحكم بعمل المعالج الدقيق 8088/086 من خلال البرنامج المكتوب بلغة التجميع . حيث يستطيع المستخدم من خلال برنامجه تغيير قيم بعض الرايات ، ايقاف المعالج عن العمل او جعله في حالة انتظار حدث ما. تقسم تعليمات التحكم بالمعالج الدقيق الى ثلاث مجموعات : –

- تعليمات الرايات
- تعليمات التزامن الخارجي
- تعليمة « لا عمل » No - operation

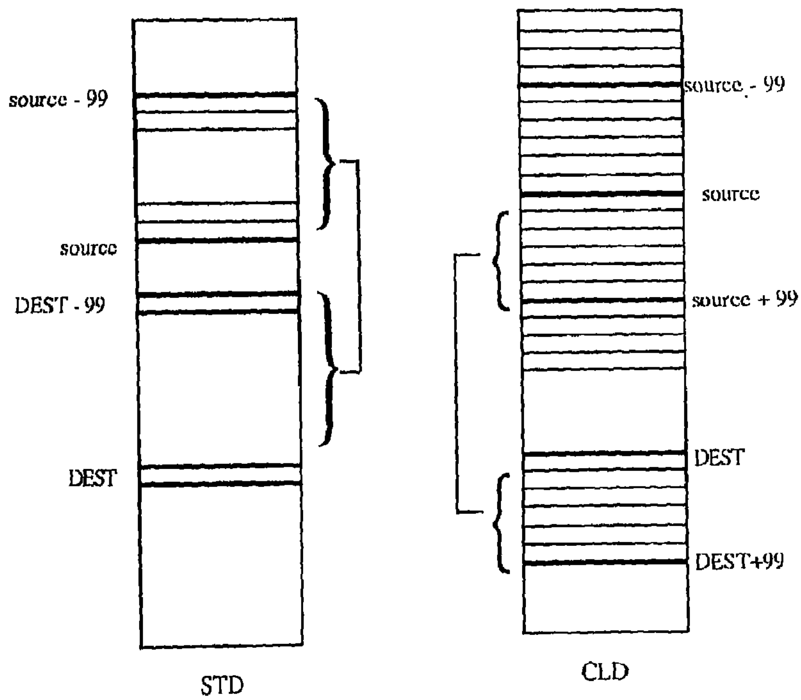
تستخدم تعليمات الرايات للتحكم في قيم الرايات كما يلي : –

التعليمة	الراية المتأثرة	قيمة الراية
STC	CF	1
CLC	CF	0
CMC	CF	يعكس الحالة
STD	DF	١
CLD	DF	0
STI	IF	1
CLI	IF	0

يتضح من الجدول ان المستخدم يستطيع التحكم بعمل المعالج الدقيق بواسطة تغيير قيم راية الحمل CF ، راية الاتجاه DF وكذلك راية الاعتراض IF . فمثلا تستخدم التعلميات STD أو CLD لوضع راية الاتجاه في حالة واحد او صفر . والتي تحدد بدورها اتجاه معالجة سلاسل الرموز strings character كما في المثال التالي : –

```
        CLD                     ; set  DF = 0
        LEA     SI, SOURCE
        LEA     DI , ES : DEST
        MOV     CX , 100        ; Byte counter
REP     MOVSB
```

فالتعليمة CLD تؤدي الى تصفير راية الاتجاه DF = 0 التي تؤدي الى زيادة واحد الى كل من SI و DI بعد كل مرة تنفذ فيها التعليمة MOVSB . وكنتيجة لتنفيذ هذه التعليمات فإن محتويات خلايا الذاكرة المائة المحصورة بين العناوين SOURCE + 99 , SOURCE ستنقل الى مواقع الذاكرة في العناوين من DEST الى DEST + 99 . ان استبدال التعليمة CLD بالتعليمة STD يؤدي الى نقل محتويات مواقع الذاكرة من SOURCE الى SOURCE - 99 الى المواقع من DEST الى DEST - 99 . يوضح الشكل التالي عملية النقل باستخدام التعليمات CLD , STD .



تستخدم التعليمات CLT , STI لوضع راية الاعتراض في حالة واحد او صفر . يستطيع المستخدم استعمال هذه التعلميات لتحديد امكانية او عدم امكانية معالجة الاعتراضات . فاذا كانت IF=1 فان المعالج الدقيق يقوم بمعالجة الاعتراضات بأنـواعها . امـا اذا كانت IF = 0 فان الاعتراضات المصنعة maskable interrupts يتم اهمالها أو تأجيل معالجتها . اما الاعتراضات غير المقنعة فلا يمكن اهمالها أو تأجيل معالجتها .

تستخدم تعليمات التزامن الخارجي External synchronization instructions لتنظيم عمل المعالج الدقيق 8086 / 8088 مع الاحداث الناتجة في ملحقات الحاسب الخارجية . فالتعليمة HLT تؤدي الى وضع المعالج الدقيق 8088 / 8086 في حالة التوقف الخامل halt state التي لا يخرج منها الاّ بأمر تصفيره rest او في حالة وصول اعتراض خارجي External interrupt . عند دخول هذه الحالة يبقى المعالج الدقيق خاملا ولا ينفذ اي عمل .

تستعمل التعليمة WAIT لوضع المعالج الدقيق في حالة التوقف النشيط . وهي حالة مشابهة للتوقف الخامل ، الا ان المعالج الدقيق عند دخوله حالة التوقف النشيط يقوم بفحص خط الدخول المسمّى TEST في فترات منتظمة طول كل منها 5 نبضات . يبقى المعالج الدقيق في حالة التوقف النشيط الى ان يتحول الخط TEST الى الحالة النشطة . يتبين من هذا ان التعليمة WAIT يمكن ان تستخدم لايقاف عمل المعالج الدقيق لغاية وصول حدث معين . فمثلا في انظمة الحاسب التي تحتوي عدة معالجات دقيقة مثل 8086 و 8087 والتي تشترك في معالجة البيانات ، يمكن استخدام تعليمة WAIT لايقاف المعالج 8086 حتى يخلص المعالج 8087 من تنفيذ العملية التي يقوم بها حاليا .

تستخدم التعليمة ESC في أنظمة الحاسب متعددة المعالجات ، حيث يمكن باستخدام تعليمة ESC في البرنامج تنفيذ تعليمات اي معالج آخر .

فمثلا ، في نظام الحاسب الذي يحتوي على المعالج 8087 بالاضافة الى المعالج 8086 يمكن ارسال تعليمات الى المعالج 8087 من خلال البرنامج المخصص للمعالج 8086

يستخدم المقطع LOCK (عبارة عن بايت واحد يسبق بة تعليمة) لمنع اي معالج دقيق غير 8086 من استعمال الناقلة BUS لفترة مساوية لزمن تنفيذ التعليمة التي يسبقها .

تستخدم التعليمة NOP (NO OPERATION) لأغراض متعددة منها : -

- تعديل البرنامج الهدفي object program باستبدال تعليمة أو اكثر بالشيفرة 90H ثم تنفيذه دون اعادة ترجمته .

- برمجة الفترات الزمنية Time - delays وغيرها .

ان تنفيذ التعليمة NOP من قبل المعالج الدقيق لا تؤدي الى اي تغيير في قيم الرايات, المسجلات او مواقع الذاكرة . ولكنها تؤدي الى التأثير على محتويات مؤشر التعليمة IP فقط .

الوحدة العاشرة

برمجة الادخال والاخراج

- مفهوم الاعتراض
- تعليمات الاعتراض
- برمجة الادخال والاخراج
- الطباعة

برمجة الادخال والاخراج

مفهوم الاعتراض

يعرف الاعتراض على انه عملية الانقطاع في تنفيذ البرنامج الرئيسي للانتقال الى تنفيذ برنامج الاعتراض الفرعي المخزن في الذاكرة الداخلية ، وذلك نتيجة عامل خارجي او داخلي . وهي عملية مشابهة لعملية استدعاء برنامج فرعي باستخدام تعليمة (CALL) حيث يتم الرجوع الى البرنامج الرئيسي بعد تنفيذ برنامج الاعتراض الفرعي.

يقوم المعالج 8086 بتنفيذ ٢٥٦ نوع من عمليات الاعتراض ، ترقم من 0 حتى 255 حسب اولويات التنفيذ . وبهذا يكون للاعتراض رقم 0 أعلى اولويه ولرقم 255 ادنى اولوية .

وتستخدم هذه الارقام للحصول على عنوان برنامج الاعتراض الفرعي بواسطة جدول مؤشر عنوان الاعتراض المبين في الشكل التالي ويحتوي هذا الجدول على ٢٥٦ مؤشر عنوان ، محتويات كل واحد منها تشير الى عنوان برنامج الاعتراض الفرعي . تبلغ سعة كل مؤشر ٤ بايت .

يخزن جدول مؤشر عنوان الاعتراض في المواقع الاولى في الذاكرة الرئيسية ويحتل المواقع من عنوان 00000 حتى 003FE .

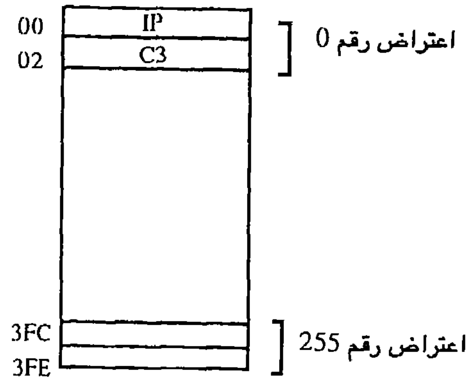
والشكل اللاحق يبين الطريقة التي يتم بواسطتها الحصول على عنوان برنامج الاعتراض الفرعي . حيث يقوم المعالج بقراءة رقم الاعتراض وتحويله الى عنوان مؤشر عنوان الاعتراض المخزن في جدول مؤشر العنوان ثم يتم نقل محتويات مؤشر العنوان الى المسجل IP , CS .

حيث يتم نقل البايت الاول والثاني الاقل اهمية من المؤشر الى IP والبايت الثالث والرابع الاكبر اهمية الى CS .

مثال : للحصول على عنوان برنامج الاعتراض الفرعي للاعتراض رقم 1AH ، نتبع الخطوات التالية :

١ -نقوم بحساب عنوان مؤشر عنوان الاعتراض في الجدول والذي يساوي . 68H = 1AH x 4

٢ – محتويات الموقع 68H يتم نقلها الى CS , IP على التوالي .



عند تنفيذ عملية الاعتراض ، بالاضافة الى حساب عنوان الاعتراض يقوم المعالج بالعمليات التالية:

١ – طرح 2 من SP ، ثم تخزين مسجل الرايات في ذاكرة الكومة .

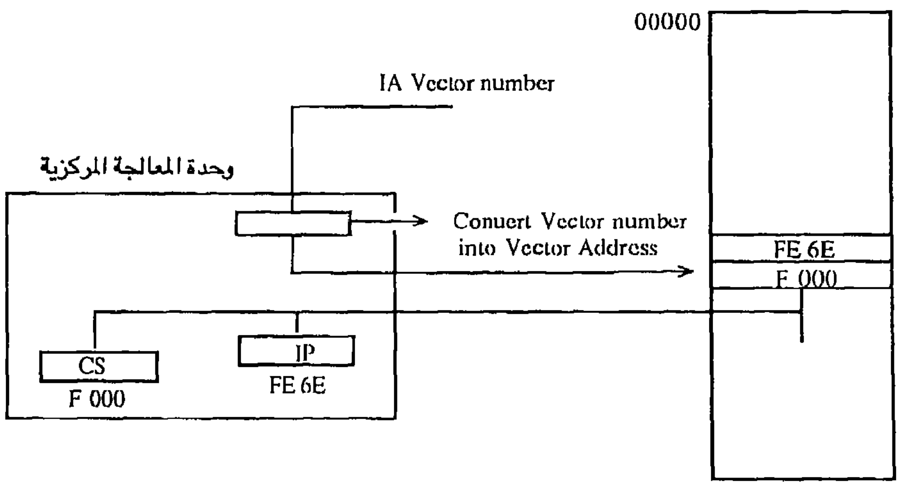
٢ – تنظيف الخانات IF , TF من مسجل الرايات .

٣ – طرح 2 من SP ، وتخزين CS في ذاكرة الكومة .

٤ – طرح 2 من SP ، وتخزين IP في ذاكرة الكومة .

٥ – اجراء عمليات برنامج الاعتراض الفرعي .

٦ – استرجاع القيم التي تم تخزينها في مسجل CS , IP ، ومسجل الرايات . والعودة الى البرنامج الرئيسي ، والى الجملة التي تلي جملة INT .



جدول مؤشر عنوان الاعتراض

تعليمات الاعتراض

الانقطاع في تنفيذ البرنامج الرئيسي يمكن ان ينشأ نتيجة اعتراض خارجي قادم من الوحدات الطرفية المرتبطة مع وحدة المعالجة المركزية ، وذلك بارسال اشارة الاعتراض على المدخل INTR للمعالج ، ويمكن ان ينشأ الانقطاع في التنفيذ نتيجة تنفيذ تعليمة الاعتراض الموجودة في البرنامج الرئيسي .

تعليمات الاعتراض

الصيغة العامة لتعليمة الاعتراض هي كما يلي :

INT int - type

حيث ان int - type تمثل رقم الاعتراض المطلوب تنفيذه . عند تنفيذ هذه التعليمة يقوم المعالج بتخزين محتويات المسجل IP والمسجل CS في ذاكرة الكومة على التوالي ، ثم بجعل الخانة TF , IF في مسجل الرايات مساوية لصفر ، وتحميل عنوان برنامج الاعتراض الفرعي في CS , IP

برمجة الادخال والاخراج

في هذا الباب سنقوم بدراسة برمجة الادخال والاخراج باستخدام تعليمات الاعتراض INT 21H و INT 10H .

باستخدام تعليمة INT 10H يمكن للمعالج ان يقوم بجميع عمليات معالجة الشاشة ولوحة المفاتيح ، حيث تقوم هذه التعليمة بنقل التنفيذ مباشرة الى نظام الادخال والاخراج الاساسي BIOS المخزن في الذاكرة الداخلية ROM

اما INT 21H فسوف نستخدمها في عمليات ادخال البيانات عن طريق لوحة المفاتيح ، وعمليات اخراج البيانات على الشاشة وتنفيذ هذه التعليمة يؤدي الى انتقال السيطرة الى نظام التشغيل DOS

- عمليات معالجة الشاشة باستخدام INT 10H

١ - عملية تحريك المؤشر على الشاشة .

يمكن تصور الشاشة بشبكة من المواقع المعنونة ، ويمكن للمؤشر الحركة الى اي موقع من تلك المواقع باعطاء احداثيات الموقع المكونة من رقم السطر ورقم العمود . في الشاشة العادية يوجد ٢٥ سطرا ترقم من 0 حتى 24 ، ٨٠ عمودا ترقم من 0 حتى 79 .

الموقع	الاحداثيات	
	رقم العمود	رقم السطر
الزاوية اليسرى العليا من الشاشة	00	00
وسط الشاشة	39/40	12
الزاوية اليسرى السفلية	00	24
الزاوية اليمنى السفلى	79	24

مثال : – لتحريك المؤشر الى منتصف الشاشة ، نقوم بتحميل احداثيات منتصف الشاشة في DX ، بحيث ان DL يحتوي على رقم السطر وDH على رقم العمود . وتحميل AH برقم الوظيفة المطلوبة من الاعتراض رقم 10H ويحمل BH برقم الصفحة وعلى هذا الاساس يكون الاجراء كما يلي

```
MOV   AH , 2
MOV   BH , 0
MOV   DL,
MOV   DX,
10H   INT
```

مثال : لعرض النص التالي على الشاشة

WHAT IS YOUR NAME

نقوم بتعريف ذلك النص في الذاكرة كما يلي

```
X   DB  'WHAT IS YOUR NAME' , $
```

ومن ثم نستخدم التعليمات الخاصة بعرض البيانات بتحميل AH برقم الوظيفة المطلوبة من الاعتراض 21H ، وفي هذه الحالة AH سوف يحتوي على رقم 9 ، ويحمل المسجل DX بعنوان بداية منطقة الذاكرة المطلوب عرضها ، ويستخدم الرمز$ للاشارة الى نهاية المنطقة المطلوب عرضها فتكون العملية كما يلي

```
MOV   AH , 9
LEA   DX , X
INT   21H
```

– ادخال البيانات عن طريق لوحة المفاتيح باستخدام INT 21H

للقيام بهذه العملية نقوم بتعريف منطقة في الذاكرة مخصصة لاستقبال البيانات المدخلة والتي يتم فيها تعريف اكبر عدد للرموز التي يمكن ان تتكون منها البيانات المدخلة ، وكذلك تعريف الطول الفعلي للبيانات المدخلة ، وتعريف المنطقة المخصصة للرموز المدخلة .

وعلى هذا الاساس فان منطقة الادخال تتكون من ثلاثة اجزاء وهي

١ – البايت الاول يخصص لاكبر عدد للرموز

٢ – البايت الثاني يحتوي على الطول الفعلي للبيانات المدخلة

٣ – منطقة تخزين الرموز المدخلة والتي تتكون من عدة مواقع .

مثال – تعريف منطقة ذاكرة مخصصة لادخال بيانات لا يتجاوز طولها 15 رمزا

```
X PAR     LABEL  BYTE
X LEN     DB 15
X ALEN    DB  ?
X FLD     DB  15 DUP ('  ')
```

باختصار يمكن القول بأنه للقيام بعملية الاعتراض INT 10H من اجل تحريك المؤشر نقوم بتهيئة المسجلات اللازمة لذلك وهي AH , BH , DX حيث ان :

AH – يحتوي على رقم الوظيفة المطلوبة من الاعتراض 10H

BH – رقم الشاشة المطلوب معالجتها .

DL – رقم العمود للموقع .

DH – رقم السطر للموقع .

٢ – عملية مسح الشاشة

لاجراء عملية مسح الشاشة باستخدام INT 10H نقوم بتهيئة المسجلات التالية قبل تنفيذ التعليمة INT 10H ، والمسجلات هي

AH – يحمل رقم الوظيفة المطلوبة من الاعتراض 10H

CX – احداثيات منطقة بداية المسح

DX – احداثيات منطقة نهاية المسح

BH – تحديد رقم الصفحة

وعلى هذا الاساس يكون اجراء مسح الشاشة كما يلي

```
MOV   AH , 06
MOV   BH , 07
MOV   CX , 00
MOV   DX , 184F
INT    10H
```

برمجة عمليات الادخال والاخراج
باستخدام INT 21H

١ – عملية عرض البيانات على الشاشة

تقوم هذه العملية بعرض محتويات منطقة ذاكرة على الشاشة .وللقيام بهذه العملية نقوم بتحديد رقم الوظيفة المطلوبة من الاعتراض رقم 21H وتحميلها في AH ، وبتحميل DX بالعنوان الفعلي لمنطقة الذاكرة المطلوب عرضها ، ويستخدم الرمز $ للاشارة الى نهاية المنطقة المطلوب عرضها .

ولاجراء عملية ادخال البيانات الى تلك المنطقة لا بد من تحميل AH برقم وظيفة الاعتراض INT 21H وتحميل DX بالعنوان الفعلي للمنطقة المخصصة للادخال .

```
MOV   AH , 10
MOV   DX , offset XPAR
INT    21H
```

عند تنفيذ هذا الاجراء سوف ينتظر الجهاز من المبرمج ان يدخل البيانات المطلوبة والتي يجب ان لا تتجاوز اكبر عدد ممكن للرموز المعرف في البايت الاول لمنطقة الادخال (في المثال عدد الرموز = ٢٠)

يستطيع المبرمج ان يستخدم مفتاح الادخال للاشارة الى نهاية سلسلة الرموز التي ادخلها اذا كانت اقل من اكبر عدد مسموح به .

فاذا قمنا على سبيل المثال بادخال الاسم AHMAD فان منطقة الادخال سوف تكون بعد الادخال كما يلي

|15|5|A|H|M|A|D|#|1...

مثال : اكتب برنامج بلغة اسمبلي يقوم بقراءة اسم من لوحة المفاتيح ثم عرضه على الشاشة على شكل قطري .

```
STACK    SEGMENT STACK PARA 'SATCK'
                 DW 1000 DUP (?)
STACK        ENDS
DAT              SEGMENT PARA 'DATA'
XPAR             LABEL BYTE
XLEN             DB 10
XALEN    DB  ?
XAREA        DB  10 DUP (' '),'$'
MESS             DB 'ENTER YOUR NAME','$'
DAT              ENDS
COD              SEGMENT PARA 'CODE'
BEG      PROC    FAR
                 ASSUME CS:COD,DS:DAT,SS:STACK,ES:NOTHING
                 PUSH DS
```

```
                SUB AX,AX
                PUSH AX
                MOV AX,DAT
                MOV DS,AX
                CALL CLSS
                CALL DM             ;DISPLAY MESSAGE
                MOV DX,0000
                CALL SETC           ;SET CURSOR TO COLUMN 0,AND ROW 0
                CALL ENTD           ;ACCEPT NAME FROM KBD
                CALL CLSS           ;CLEAR SCREEN
                MOV DX,0000
LL:          CALL SETC
                PUSH DX             ;SAVE DX INTO THE STACK
                CALL DISPN          ;DISPLAY ENTERED NAME
                POP DX              ;READ DX FROM TOP OF STACK
                ADD DL,3
                ADD DH,1
                CMP DH,24
                JB LL
                RET
BEG             ENDP
;       CLEAR SCREEN
;  ------------------
CLSS            PROC
                MOV AL,0
                MOV AH,6            ;REQUEST CLEAR SCREEN
                MOV BH,30           ;ATTRIBUTE BYTE
                MOV CX,0            ;UPER CORNER
                MOV DX,184FH
                INT 10H
                RET
CLSS            ENDP
;       SET CURSOR
;       ------------
SETC            PROC
                MOV AH,2
                MOV BH,00
                INT 10H
                RET
SETC            ENDP
;       ACCEPT DATA
;       -------------------
ENTD            PROC NEAR
                LEA DX,XPAR
                MOV AH,10
                INT 21H
                RET
ENTD            ENDP
;       DISPLAY NAME
;       ------------------
DISPN   PROC  NEAR
                LEA DX,XAREA
                MOV AH,9
                INT 21H
                RET
DISPN        ENDP
;       DISPLAY NAME
;       ---------------
DM              PROC
                MOV AH,9
                LEA DX,MESS
                INT 21H
                RET
DM              ENDP
COD             ENDS
                END
```

عمليات معالجة الشاشة

يمكن استخدام اساليب عرض مختلفة للبيانات على الشاشة العادية والملونة وذلك بتحميل المسجل BL ببايت مواصفات الرموز المعروضة على الشاشة وتحميل المسجل AH برقم الوظيفة المطلوبة من برنامج الاعتراض رقم 10H .

تزود الشاشة العادية بذاكرة تصل سعتها الى ٤ ك . ب ، ٢ك . ب منها تستخدم لتخزين الرموز التي تظهر على الشاشة ، وتتسع لـ (٢٥×٨٠) رمزا ، وهذا هو عدد الرموز التي يمكن عرضها على الشاشة في نفس اللحظة اما بقية الذاكرة فتستخدم لتخزين بايت مواصفات العرض لكل رمز من الرموز على الشاشة .

اما الشاشات الملونة فيمكنها العمل كشاشة ملونة للرسم وكتابة النصوص المختلفة عليها ، وكذلك يمكن استخدامها كشاشة عادية. تزود مثل هذه الشاشات بذاكرة تصل سعتها ال ١٦ ك . ب تستخدم هذه الذاكرة لتخزين رموز ومواصفات اربع صفحات ، كل صفحة تتسع لـ (٢٥×٨٠) رمزا مع المواصفات لكل رمز .

بايت مواصفات الرموز المعروضة على الشاشة

يستخدم هذا البايت لتحديد خصائص كل رمز سيتم عرضه على الشاشة ويحتوي على ٨ خانات ثنائية ، الاربع خانات الاقل وزنا تصف لون الرمز وشدة اضاءته ، والاربع خانات الاعلى وزنا تصف لون الخلفية التي يعرض عليها الرمز والاضاءة المتقطعة ، والشكل التالي يبين تلك الخانات

BL	R	G	B	I	R	G	B
7	6	5	4	3	2	1	0

الحرف في الخانة 0 يمثل اللون الازرق ، والحرف G في الخانة ١ يمثل اللون الاخضر ، و R يمثل اللون الاحمر . أما الخانة الممثلة بالحرف I فهي تشير الى شدة اضاءة الرمز ، والخانة BL تمثل الاضاءة المتقطعة لعرض رمز معين على الشاشة حسب مواصفات معينة ، يتم تحميل بايت المواصفات في المسجل BL قبل عملية استدعاء برنامج الاعتراض رقم 10H

بالاضافة الى عملية مسح الشاشة ، وتحريك المؤشر يقوم الاعتراض 10H بالعمليات التالية

١ - عملية اختيار نموذج شاشة العرض - تتم هذه العملية بتحميل المسجل AH بصفر والذي يشير الى رقم الوظيفة المطلوبة من الاعتراض 10H وتحميل AL برقم

نموذج شاشة العرض والذي يمكن اختياره من الجدول المبين في الاسفل

شاشة ابيض اسود 40x25	00
شاشة ملونة 40x25	01
شاشة ابيض اسود80X25	02
شاشة ملونة 80x25 /١٦ لون	03
شاشة ملونة للرسم 320x200 / ٤ الوان	04
شاشة ابيض/اسود للرسم 310x200	05
شاشة ابيض/اسود للرسم 640x200	06
شاشة ابيض/اسود 80x25	07
(EGA) شاشة للرسم/١٦ لون 320x200	0D
(EGA) شاشة للرسم/١٦ لون 640x200	0E
(EGA) شاشة ابيض/اسود 640x200	0F
شاشة ملونة/١٤ لون 640x350	10

٢- عملية ازاحة الشاشة للأعلى بعدد معين من الاسطر AH = 6 (Scrolling up)

باستخدام هذه العملية يتم ازاحة اي جزء من الشاشة للاعلى بمقدار عدد الاسطر التي يتم تحديدها في المسجل AL ، الامر الذي يؤدي الى اختفاء الاسطر العلوية وظهور اسطر جديدة في اسفل الشاشة يتم تحديد جزء الشاشة المطلوب ازاحته للاعلى باعطاء احداثيات ذلك الجزء في DX , CX

مثال : - لتحريك الشاشة للاعلى بمقدار سطر واحد نكتب الاجراء التالي

```
MOV  AH , 06
MOV  AL , 01          عدد الاسطر
MOV  CX , 0           احداثيات بداية المنطقة
MOV  DX , 1884FH      احداثيات نهاية الشاشة
MOV  BH , 07          بايت المواصفات
INT  104
```

وفيما يلي بعض الامثلة على محتويات ذلك البايت وتأثيرها على الشاشة العادية عند استدعاء برنامج الاعتراض 10H

محتويات BL	التــاثيــر
00	شاشة سوداء
07 Hex	ابيض /اسود عادي
87 Hex	ابيض / اسود اضاءة متقطعة
OF Hex	ابيض / اسود اضاءة شديدة
70 Hex	اسود / ابيض اضاءة معكوسة
F0 Hex	أسود / ابيض اضاءة معكوسة متقطعة

في الشاشة الملونة هناك ٣ الوان اساسية ويمكن الخلط بين تلك الالوان لاعطاء 8 الوان من ضمنها الاسود والابيض

بواسطة الاربع خانات الاقل وزنا من بايت المواصفات يمكن توليد ١٦ لونا للرمز الذي سيتم عرضه كما هو مبين في الشكل ، اما الخانات الباقية فهي تولد 8 الوان للخلفية التي سيعرض عليها الرمز

اللون	I R G B
اسود	0 0 0 0
ازرق	0 0 0 1
اخضر	0 0 1 0
سماوي	0 0 1 1
احمر	0 1 0 0
وردي	0 1 0 1
بني	0 1 1 0
ابيض	0 1 1 1
رمادي	1 0 0 0
ازرق شديد الاضاءة	1 0 0 1
اخضر شديد الاضاءة	1 0 1 0
سماوي شديد الاضاءة	1 0 1 1
احمر شديد الاضاءة	1 1 0 0
وردي شديد الاضاءة	1 1 0 1
اصفر	1 1 1 0
ابيض ناصع	1 1 1 1

٣ – عملية ازاحة الشاشة للاسفل – باستخدام هذه العملية يتم ازاحة اي جزء من الشاشة للاسفل مقدار عدد معين من الاسطر التي يتم تحديدها في المسجل AL ، الامر الذي يؤدي الى اختفاء الاسطر السفلية وظهور اسطر جديدة فارغة في اعلى الصفحة .

رقم الوظيفة هو 7 يحمل في AH

٤ – عملية اظهار رمز معين في الموقع الحالي للمؤشر AH = 9

الرمز المطلوب عرضه يتم تحميله في المسجل AL ، ورقم الوظيفة في AH وعدد مرات اظهار الرمز يتم تحميله في CX

مثال : لاظهار الرمز * على الشاشة خمس مرات نكتب الاجراء التالي

```
MOV  AH , 9     -            رقم الوظيفة
MOV  AL , '*'   -  الرمز المطلوب اظهاره
MOV  BH , 0     -            رقم الصفحة
MOV  BL , 14    -            احمر / ازرق
MOV  CX , 5     -              عدد المرات
INT  10H
```

تنفيذ هذا الاجراء يؤدي الى عرض خمس نجوم حمراء على خلفية زرقاء والذي يوضح ذلك محتوى مسجل BL

يجب ان نلاحظ ان هذه العملية لا تؤدي الى تحريك المؤشر .

مثال :

يقوم البرنامج المبين ادناه بقراءة اسم من لوحة المفاتيح لا تزيد رموزه عن ٢٠ رمزا ، واظهار هذا الاسم على الشاشة باضاءة متقطعة بلون اخضر على خلفية وردية اللون والخطوات التالية تبين عمل البرنامج.

١ – استدعاء البرنامج الفرعي ENTD ، والذي يقوم بادخال الاسم من لوحة المفاتيح الى منطقة الذاكرة المخصصة لذلك وهي NFLD

٢ – اظهار الاسم الموجود في موقع الذاكرة NFLD على الشاشة باضاءة متقطعة ، ويتم ذلك باستدعاء البرنامج الفرعي KK الذي يقوم بدوره باستدعاء البرنامج الفرعي DN الذي يظهر رمزا واحدا فقط من الاسم في كل مره يتم تنفيذه فيها .

```
STACK   SEGMENT STACK PARA 'SATCK'
                DW 1000 DUP (?)
 STACK        ENDS
 DAT            SEGMENT PARA 'DATA'
 NPAR   LABEL BYTE
 NLEN   DB 20
NALEN         DB ?
NFLD          DB 20 DUP(' ')
ROW             DB 00
COL           DB 00
MESS            DB 'ENTER YOUR NAME',13,10,'$'
DAT             ENDS
COD             SEGMENT
 BEG            PROC  FAR
                ASSUME CS:COD,DS:DAT,SS:STACK,ES:NOTHING
                PUSH DS
                SUB AX,AX
                PUSH AX
                MOV AX,DAT
                MOV DS,AX
                CALL CLSS         ;CLEAR THE SCREEN
                MOV COL,00
                MOV ROW,02
                CALL SETC         ;SET CURSOR TO COL 00 &ROW 02
                CALL DM           ;DISPLAY MESSAGE
                CALL ENTD         ;ACCEPT THE NAME FROM KBD
                CALL KK           ;DISPLAY THE NAME AS LIGHT GREEN ON MAGE
                RET
 BEG          ENDP
;           CLEAR SCREEN
;         ----------------------
CLSS          PROC
              MOV AL,0
                MOV AH,06         ;REQUEST CLEAR SCREEN
                MOV BH,02         ;COLOR ATTRIBUTE (GREEN ON BLACK)
                MOV CX,0          ;UPPER LEFT ROW/COL
                MOV DX,184FH      ;LOWER RIGHT CORNER ROW/COL
                INT 10H           ;EXIT TO BIOS
                RET
 CLSS         ENDP
              DISPLAY MESSAGE
          --------------------
 DM             PROC
                LEA DX,MESS       ;LOAD EFFECTIVE ADDRESS OF MESS INTO DX
                MOV AH,9          ;REQUIST DISPLAY
                INT 21H           ;EXIT TO DOS
                RET
 DM             ENDP
              ACCEPT NAME FROM KBD
          ----------------------------
 ENTD           PROC
                LEA DX,NPAR       ;LOAD EFFECTIVE ADDRESS OF NPAR INTO DX
                MOV AH,10         ;REQUEST INPUT
                INT 21H
                RET
 ENTD           ENDP
```

```
          DISPLAY NAME AS LIGHT GREEN ON MAGENTA WITH BLINKING
         ------------------------------------------------------
KK             PROC
               LEA SI,NFLD
               ADD ROW,2
               ADD COL,0
LL:         CALL SETC
               MOV BL,0DAH
               CALL DN          ;DIPLAY CHARACTER PROCEDURE
               INC SI           ;NEXT CHARACTR IN THE NAME
               INC COL          ;NEXT COLUMN ON THE SCREEN
               DEC NALEN        ;DECREMENT NAME LENGTH
               JNZ LL           ; GO TO LL LABEL IF NALEN IS NOT ZERO
               RET
KK          ENDP
          SET CURSOR PROCEDURE ROW/COL
         ------------------------------
SETC           PROC
               MOV AH,02        ;REQUEST SET CURSOR
               MOV BH,00        ;PAGE #0
               MOV DH,ROW       ;MOVE ROW NUMBER TO DH
               MOV DL,COL       ;MOV COL NUMBER TO DL
               INT 10H          ; EXIT TO BIOS
               RET
SETC           ENDP
         DISPLAY CHARACTR
         ------------------
DN             PROC
               MOV AH,9         ;REQUEST DISPLAY
               MOV AL,[SI]      ;GET NAME CHARACTER
               MOV BH,00        ;PAGE #0
               MOV CX,01        ;DISPLAY ONE CHARACTER
               INT 10H          ;EXIT TO BIOS
               RET
DN             ENDP
COD            ENDS
               END
```

مثال : اكتب برنامج يقوم بقراءة عددين صحيحين من لوحة المفاتيح ، وايجاد مجموعهما واظهار المجموع على الشاشة

```
STACK   SEGMENT STACK PARA 'STACK'
                DW 1000 DUP (' ')
STACK   ENDS
DAT             SEGMENT PARA 'DATA'
XPAR            LABEL BYTE
XLEN          DB  6
XALEN   DB  ?
XAREA   DB  6 DUP (' '),13,10,'$'
YPAR            LABEL  BYTE
YLEN            DB  6
```

```
YALEN   DB ?
YAREA   DB 6 DUP (' '),13,10,'$'
MESS            DB 'the sum of two integers is',13,10,'$'
MESS1   DB 'ENTER THE FIRST  NUMBER',13,10,'$'
MESS2   DB 'ENTER THE SECOND NUMBER ',13,10,'$'
MULT            DW 0001H
TENW            DB 10
XBIN            DW ?
YBIN            DW ?
ZBIN            DW ?
ASCV            DB 6 DUP (30H),13,10,'$'
BIN             DW ?
COUN    DB 00
COL             DB 01
ROW             DB 01
DAT             ENDS
COD             SEGMENT PARA 'CODE'
BEG             PROC  FAR
                ASSUME CS:COD,DS:DAT,SS:STACK,ES:NOTHING
                PUSH  DS
                SUB AX,AX
                PUSH AX
                MOV AX,DAT
                MOV DS,AX
             CALL CLSS  ;CLEAR SCREEN
                CALL CLR         ;CLEAR INPUT AREA
             CALL ENTD  ;ACCEPT NUMBERS FROM KBD
                CALL CONV1       ;CONVERT THE FIRST NUMBER FROM ASCII
                                 ;TO BINARY
                MOV AX,BIN       ;SAVE THE RESULT OF CONV1 IN AX
                MOV XBIN,AX      ;SAVE THE RESULT OF CONVESION IN XBIN
                MOV AX,0
                CALL CONV2       ;CONVERT THE SECOND NUMBER  FROM ASCII
                                 ;TO BINARY
                MOV AX,BIN
                MOV YBIN,AX      ;SAVE THE RESULT IN YBIN
                MOV AX,0
                CALL PROCESS     ;FIND THE SUM & STORE IT IN ZBIN
                CALL ACON        ;CONVERT ZBIN INTO ASCII CODE
                CALL DM          ;PRINT MESSAGE
                CALL BL1         ;PRINT THE SUM AS LIGHT GREEN ON
                                 ;MAGENTA ,BLINKING
                RET
BEG             ENDP
;      ACCEPT NUMBERS FOM KBD
;        -------------------------
ENTD            PROC
                MOV ROW,5
                MOV COL,5
                CALL SETC        ;CALL SET THE CURSOR
                LEA DX,MESS1     ;MOV EFFECTIVE ADDRES OF MESS1 INTO DX
                CALL DISP
                MOV DX,0000
                LEA DX,XPAR
                MOV AH,10
                INT 21H
                ADD ROW,2
```

```
                ADD COL,0
                CALL SETC
                MOV DX,0000
                LEA DX,MESS2
                CALL DISP
                LEA DX,YPAR
                MOV AH,10
                INT 21H
                ADD ROW,2
                ADD COL,0
                CALL SETC
                RET
ENTD            ENDP
;        CONVERT FIRST NUMBER FROM ASCII TO BINARY
;       ------------------------------------------
CONV1   PROC
                MOV CX,10
                LEA SI,XAREA-1
                MOV BH,00
                MOV BL,XALEN
B20:            MOV AL,[SI+BX]
                AND AX,000FH
                MUL MULT
                ADD BIN,AX
                MOV AX,MULT
                MUL CX
                MOV MULT,AX
                DEC BX
                JNZ B20
                RET
CONV1   ENDP
;       CONVERT SECOND NUMBER FROM ASCII TO BINARY
;    -------------------------------------------
CONV2   PROC
                MOV MULT,0001
                MOV BIN,0000
                MOV CX,10
                LEA SI,YAREA-1
                MOV BH,00
                MOV BL,YALEN
B201:
                MOV AL,[SI+BX]
                AND AX,000FH
                MUL MULT
                ADD BIN,AX
                MOV AX,MULT
                MUL CX
                MOV MULT,AX
                DEC BX
                JNZ B201
                RET
 CONV2  ENDP
;       THE SUM OF TWO NUMBERS
;    -----------------------
PROCESS      PROC
                MOV DX,XBIN
                MOV AX,YBIN
```

```
                ADD AX,DX
                MOV ZBIN,AX
                RET
PROCESS      ENDP
;        CONVERT THE RESULT FROM BINARY INTO ASCII CODE
;        -----------------------------------------------
ACON            PROC
                MOV CX,0010
                LEA SI,ASCV+5
                MOV AX,ZBIN
C20:
                CMP AX,0010
                JB C30
                XOR DX,DX
                DIV CX
                OR DL,30H
                MOV [SI],DL
                DEC SI
                JMP C20
C30:
                OR AL,30H
                MOV [SI],AL
                RET
ACON            ENDP
;        DISPLAY MESSAGE
;        ----------------
DISP            PROC
                MOV AH,9
                INT 21H
                MOV ASCV[SI],30H
                RET
;        CLEAR SCREEN
;        -------------
DISP            ENDP
;        CLEAR INPUT AREA
;        ----------------
CLR             PROC
                MOV CX,6
                MOV SI,0
L:              MOV XAREA[SI],20H
                MOV YAREA[SI],20H
                INC SI
                LOOP L
                RET
CLR             ENDP
CLSS            PROC
                MOV AL,0
                MOV AH,06
                MOV BH,07
                MOV CX,0
                MOV DX,184FH
                INT 10H
                RET
CLSS            ENDP
;        DISPLAY MESSAGE
;        ---------------
DM              PROC
```

```
            MOV AH,9
            LEA DX,MESS
            INT 21H
            RET
DM          ENDP
;       DISPLAY THE RESULT ,BLINKING
;       ----------------------------
BL1         PROC
            MOV COUN,6
            LEA SI,ASCV
            INC ROW
            INC COL
KK:         CALL SETC
            MOV BL,ODAH
            CALL BLIN
            INC SI
            INC COL
            DEC COUN
            JNZ KK
            RET
BL1         ENDP
;       DISPLAY CHARACTER
;       ------------------
BLIN        PROC
            MOV AH,9
            MOV AL,[SI]
            MOV BH,00
            MOV CX,1
            INT 10H
            RET
BLIN        ENDP
;       SET CURSOR
;       --------------
SETC        PROC
            MOV AH,02
            MOV BH,00
            MOV DH,ROW
            MOV DL,COL
            INT 10H
            RET
SETC        ENDP
COD         ENDS
            END
```

قبل البدء بتوضيح عمل البرنامج لا بد من ان نشير الى ان البيانات التي يتم ادخالها عن طريق لوحة المفاتيح تكون ممثلة بنظام الشيفرة الامريكية المعيارية ، ولاظهار البيانات المخزنة في الذاكرة بالنظام الثنائي لا بد من تحويلها الى المكافىء بالشيفرة المعيارية الامريكية.

البرنامج يقوم بالخطوات التالية

١ - قراءة العدد الاول والعدد الثاني من لوحة المفاتيح باستدعاء البرنامج الفرعي ENTD ، الذي يقوم بادخال قيمة العدد الاول والعدد الثاني ممثلة بالشيفرة الامريكية

المعيارية الى منطقة الذاكرة المخصصة لذلك وهي للعدد الاول XAREA وللثاني YAREA

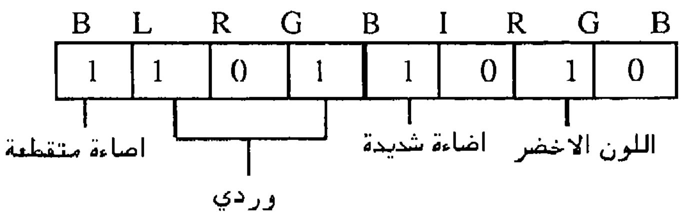
٢ – تحويل العدد الاول الممثل بالشيفرة الامريكية المعيارية والمخزن في XAREA الى مكافئه الثنائي وتخزين النتيجة (المكافىء الثنائي) في موقع الذاكرة XBIN ، ويتم ذلك باستدعاء البرنامج الفرعي CONV1 .

٣ – تحويل العدد الثاني الممثل بالشيفرة الامريكية المعيارية والمخزن في موقع الذاكرة YAREA الى مكافئة الثنائي ، وتخزين النتيجة (المكافىء الثنائي) في موقع الذاكرة YBIN . ويتم ذلك باستدعاء البرنامج الفرعي CONV2 .

٤ – جمع محتويات موقع الذاكرة YBIN مع XBIN ، باستدعاء برنامج جمع محتويات موقع الذاكرةYBIN مع XBIN وتخزين النتيجة في موقع الذاكرة ZBIN .

٥ – تحويل النتيجة المخزنة في ZBIN والممثلة بالنظام الثنائي الى مكافئة الممثل بالشيفرة الامريكية المعيارية . ويتم ذلك باستدعاء البرنامج الفرعي ACON ، وذلك من اجل عملية اظهار النتيجة على الشاشة .

٦ – استدعاء برنامج اظهار النتيجة الفرعي BLI والذي يقوم باظهار النتيجة بلون اخضر على خلفية وردية اللون وباضاءة متقطعة ويتم تحديد هذه الالوان في المسجل BL والذي يحتوي على بايت مواصفات الرمز المطلوب اظهاره على الشاشة ، ويحتوي BL في هذه الحالة 0DAH



الطباعة

PRINTING

تستخدم الالات الطابعة لاستقبال البيانات من الكمبيوتر وعرضها بشكل مقروء وعند الحديث عن الطباعة لا بد مراعاة استجابة الالة الطابعة لتقبل التعليمات الخاصة من قبل وحدة المعالجة كقلب الصفحة وبداية الطباعة في صفحة جديدة او القفز عن سطر اضافة الى امكانية وحدة المعالجة في التعرف على حالة الالة الطابعة كخلوها من الورق او كونها مشغولة او غير موصلة بالتيار الكهربائي (OFF) تختلف الالات الطابعة عن بعضها

البعض فبعض الطابعات يتقبل البيانات بشكل متوازي (بايت بايت) او بشكل تتابعي (بت بت) وبعضها قد يحتوي على وحدة تخزين قد تستطيع تخزين اعداد كبيرة من البايت للاسراع في عملية الطباعة .

ولاجراء عملية الطباعة لا بد من تنفيذ الخطوات التالية

- استصدار امر الطباعة وذلك بتخزين (40H) في AH .
- حجز الالة الطابعة وذلك بتخزين (04) في BX .
- استخدام المسجل CX كعداد بحيث يتضمن قيمة مساوية لعدد الاحرف المراد طباعتها .
- استخدام الاعتراض (21H) .

وعند اجراء عملية الطباعة لا بد من مراعاة بعض الرموز الخاصة المستخدمة ومن اهمها

للقفز الى سطر جديد   OAH Line feed

للانتقال الى صفحة جديدة   OCH Form feed

ODH Carring return (return to left margin)

للانتقال الى بداية سطر جديد من اليسار

فمثلا لطباعة الترويسة (Header)   EMPLOYEE REPORT

يتم في البداية تعريف هذه الترويسة كما يلي

HEAD DB ' EMPLOYEE REPORT ' , ODH , OAH

ولطباعة هذه الترويسة تنفذ التعليمات التالية

```
MOV   AH , 40H    ; Request output
MOV   BX , 04     ; Handle for printer
MOV   CX , 15
LEA   DX , HEADP   ; Print area
INT   21H          ; Call DOS
```

والبرنامج التالي يوضح المفاهيم الاساسية لعمليات الطباعة حيث يتضمن البرنامج ادخال مجموعة من الاسماء وطباعتها على الالة الطابعة

```
INPTM    MACRO
;ACCEPT INPUT NAMES
;------------------
                 MOV AH,40H
                 MOV BX,01
                 MOV CX,05
;REQUEST DISPLAY 5 CHARACTERS
```

```
                LEA DX,PROMPT
                INT 21H
                MOV AH,0AH
;REQUEST INPUT
                LEA DX,NAMEPAR
                INT 21H
                ENDM
PREPM    MACRO
                LOCAL LL20
;PREPARE FOR PRINTING
;-------------------
                CMP LINECTR,60
                JB LL20
                PAGEM
LL20:
                MOV CH,00
                MOV CL,NAMELEN
;SET NO. OF CHARACTERS
                LEA DX,NAMEFLD
;INITIALIZE ADDRESS OF NAME
;-------------------------
                OUTM
                MOV CX,01
                LEA DX,LFEED
                OUTM
                INC LINECTR
                ENDM
PAGEM    MACRO
                LOCAL LL30
                LOCAL LL40
                LOCAL LL50
;PAGE HEADING MACRO
;-----------------
                CMP WORD PTR PAGECTR,3130H
;FIRST PAGE ?
                JE LL30
                MOV CX,01
                LEA DX,FFEED
                OUTM
                MOV LINECTR,03
LL30:
                MOV CX,36
;LENGTH OF HEADING
                LEA DX,HEADG
LL40:
                OUTM
                INC PAGECTR+1
                CMP PAGECTR+1,3AH
                JNE LL50
                MOV PAGECTR+1,30H
                INC PAGECTR
LL50:
                ENDM
OUTM            MACRO
;PRINT MACRO
;----------
```

```
                                    MOV AH,40H
                                    MOV BX,04
                                    INT 21H
                                    ENDM
                       CLRM         MACRO
                       ;CLEAR SCREEN
                       ;-----------
                                    MOV AX,0600H
                                    MOV BH,60H
                                    MOV CX,0000
                                    MOV DX,184FH
                                    INT 10H
                                    ENDM
                       CURSM   MACRO
                       ;MACRO TO SET CURSOR ROW/COL
                       ;-------------------------
                                    MOV AH,02
                                    MOV BH,00
                                    INT 10H
                                    ENDM

                            ;-------------------------------------
                            ;ACCEPT AND PRINT NAMES
                            ;-------------------------------------
0000                        CSEG          SEGMENT PARA PUBLIC 'CODE'
                                          ASSUME CS:CSEG,DS:CSEG,SS:CSEG,
                            ES:CSEG
0100                                      ORG 100H
0100  EB 40 90              BEGIN:  JMP MAIN
                            ;------------------------------
0103                        NAMEPAR LABEL BYTE
0103  14                    MAXNLEN       DB 20
0104  00                    NAMELEN DB ?
0105  0014[                 NAMEFLD DB 20 DUP (' ')
          20
                     ]

0119  4C 49 53 54 20 4F     HEADG   DB 'LIST OF NAMES          PAGE'
      46 20 4E 41 4D 45
      53 20 20 20 20 20
      20 20 20 20 20 50
      41 47 45
0134  30 31 0A 0A           PAGECTR DB '01',0AH,0AH
0138  0C                    FFEED   DB 0CH
0139  0A                    LFEED   DB 0AH
013A  01                    LINECTR DB 01


   013B  4E 41 4D 45 20 3F     PROMPT  DB 'NAME ? '
         20
                               ;--------------------------------
   0142                        MAIN          PROC NEAR
                                             CLRM
   0142  B8 0600           1                 MOV AX,0600H
   0145  B7 60             1                 MOV BH,60H
   0147  B9 0000           1                 MOV CX,0000
   014A  BA 184F           1                 MOV DX,184FH
   014D  CD 10             1                 INT 10H
```

```
                                                     PAGEM
014F  81 3E 0134 R 3130      1                       CMP WORD PTR PAGECTR,3130H
0155  74 13                  1                       JE ??0000
0157  B9 0001                1                       MOV CX,01
015A  8D 16 0138 R           1                       LEA DX,FFEED
015E  B4 40                  2                       MOV AH,40H
0160  BB 0004                2                       MOV BX,04
0163  CD 21                  2                       INT 21H
0165  C6 06 013A R 03        1                       MOV LINECTR,03
016A                         1   ??0000:
016A  B9 0024                1                       MOV CX,36
016D  8D 16 0119 R           1                       LEA DX,HEADG
0171                         1   ??0001:
0171  B4 40                  2                       MOV AH,40H
0173  BB 0004                2                       MOV BX,04
0176  CD 21                  2                       INT 21H
0178  FE 06 0135 R           1                       INC PAGECTR+1
017C  80 3E 0135 R 3A        1                       CMP PAGECTR+1,3AH
0181  75 09                  1                       JNE ??0002
0183  C6 06 0135 R 30        1                       MOV PAGECTR+1,30H
0188  FE 06 0134 R           1                       INC PAGECTR
018C                         1   ??0002:
018C                             LOOP1:
018C  BA 0000                                        MOV DX,0000     ;SET CURSOR TO 0
                                                     CURSM
018F  B4 02                  1                       MOV AH,02
0191  B7 00                  1                       MOV BH,00
0193  CD 10                  1                       INT 10H
                                                     INPTM
0195  B4 40                  1                       MOV AH,40H
0197  BB 0001                1                       MOV BX,01
019A  B9 0005                1                       MOV CX,05
019D  8D 16 013B R           1                       LEA DX,PROMPT
01A1  CD 21                  1                       INT 21H
01A3  B4 0A                  1                       MOV AH,0AH
01A5  8D 16 0103 R           1                       LEA DX,NAMEPAR
01A9  CD 21                  1                       INT 21H
                                                     CLRM
01AB  B8 0600                1                       MOV AX,0600H
01AE  B7 60                  1                       MOV BH,60H
01B0  B9 0000                1                       MOV CX,0000
01B3  BA 184F                1                       MOV DX,184FH
01B6  CD 10                  1                       INT 10H
01B8  80 3E 0104 R 00                                CMP NAMELEN,00


01BD  74 6A                                          JE LL30
                                                     PREPM
01BF  80 3E 013A R 3C        1                       CMP LINECTR,60
01C4  72 3D                  1                       JB ??0003
01C6  81 3E 0134 R 3130      2                       CMP WORD PTR PAGECTR,3130H
01CC  74 13                  2                       JE ??0004
01CE  B9 0001                2                       MOV CX,01
01D1  8D 16 0138 R           2                       LEA DX,FFEED
01D5  B4 40                  3                       MOV AH,40H
01D7  BB 0004                3                       MOV BX,04
01DA  CD 21                  3                       INT 21H
01DC  C6 06 013A R 03        2                       MOV LINECTR,03
01E1                         2   ??0004:
01E1  B9 0024                2                       MOV CX,36
01E4  8D 16 0119 R           2                       LEA DX,HEADG
```

```
01E8                           2  ??0005:
01E8   B4 40                   3             MOV  AH,40H
01EA   BB 0004                 3             MOV  BX,04
01ED   CD 21                   3             INT  21H
01EF   FE 06 0135 R            2             INC  PAGECTR+1
01F3   80 3E 0135 R 3A         2             CMP  PAGECTR+1,3AH
01F8   75 09                   2             JNE  ??0006
01FA   C6 06 0135 R 30         2             MOV  PAGECTR+1,30H
01FF   FE 06 0134 R            2             INC  PAGECTR
0203                           2  ??0006:
0203                           1  ??0003:
0203   B5 00                   1             MOV  CH,00
0205   8A 0E 0104 R            1             MOV  CL,NAMELEN
0209   8D 16 0105 R            1             LEA  DX,NAMEFLD
020D   B4 40                   2             MOV  AH,40H
020F   BB 0004                 2             MOV  BX,04
0212   CD 21                   2             INT  21H
0214   B9 0001                 1             MOV  CX,01
0217   8D 16 0139 R            1             LEA  DX,LFEED
021B   B4 40                   2             MOV  AH,40H
021D   BB 0004                 2             MOV  BX,04
0220   CD 21                   2             INT  21H
0222   FE 06 013A R            1             INC  LINECTR
0226   E9 018C R                             JMP  LOOP1
0229                              LL30:
0229   B9 0001                               MOV  CX,01
022C   8D 16 0138 R                          LEA  DX,FFEED
                                             OUTM
0230   B4 40                   1             MOV  AH,40H
0232   BB 0004                 1             MOV  BX,04
0235   CD 21                   1             INT  21H
0237   C3                                    RET
0238                              MAIN       ENDP
0238                              CSEG       ENDS
                                             END   BEGIN
```

هذا ويمكن استخدام برمجيات الادخال والاخراج (BIOS) لاجراء عملية الطباعة باستخدام الاعتراض (17H) واعتمادا على القيمة المخزنة في AH يتم تنفيذ الوظيفة اللازمة ومن هذه الوظائف يمكن ادراج ما يلي :

١ - طباعة حرف واحد مع امكانية اختيار آلة طابعة معينة (لغاية ٣ الات طابعة) ويتم تنفيذ هذا من خلال تخزين (0) في AH كما يلي :

```
MOV   AH,00     ; Request print
MOV   AL,Char   ; Character to be printed
MOV   DX,00     ; Select printer no . 0
INT   17H       ; Call BIOS
```

٢ – تحضير منفذ الادخال والاخراج (Initialize the printer port) وذلك بتخزين القيمة (10) في AH واختيار وحدة الطباعة من خلال تحديد رقمها في المسجل DX كما يلي

```
MOV   AH,01
MOV   DX,00
INT   17H
```

٣ – قراءة حالة منفذ الالة (Read printer port status) ويتم هذا من خلال تخزين الرقم 02 في AH ثم تحديد رقم وحدة الطباعة وفحص الطابعة للتأكد من استعدادها لتنفيذ عملية الطباعة وتتم هذه العمليات كما يلي

```
MOV   AH,02
MOV   DX,00
INT   17H
TEST  AH,00101001B  ; Ready ?
JNZ   ERRORMSG
```

والغرض من (١) و (٢) هو التأكد من مدى استعداد الطابعة لاستقبال البيانات وطباعتها وذلك من خلال تحديد حالة الالة الطابعة والتي بدورها تؤثر على البت الموجودة في AH كما يلي

البت	الحالة
0	Time out
3	I/O error
4	Selected
5	out of paper
6	Acknowledged from printer
7	Not busy

الوحدة الحادية عشرة

البرمجة الميكروية

- البرمجة الجزئية
- التداخل
- مكتبة البرامج الجزئية
- التعليمات الشرطية

البرمجة الجزئية (الميكروية)
MACRO PROGRAMMING

في بعض الاحيان قد يتطلب الامر تكرار مجموعة من التعليمات في البرنامج مما يؤدي بدوره الى زيادة عدد التعليمات وزيادة احتمال الوقوع في الاخطاء بالاضافة الى زيادة في الجهد اللازم لادخال هذه التعليمات لتنفيذها وللتخلص من هذا توفر لغة التجميع امكانية تجميع التعليمات المتكررة في برنامج جزيء يمكن استدعاؤه عند الحاجة اليه .

وبهذا يمكن تعريف البرنامج الجزئي (MACRO) على انه مجموعة من التعليمات يمكن الرجوع اليها في البرنامج الرئيسي عند الحاجة اليها بدون الحاجة الى تكرارها وهذا بدوره يؤدي الى

- تقليل عدد التعليمات المدخلة .
- تجزئة البرنامج الى فقرات سهلة القراءة والفهم
- تقليل عدد الاخطاء الناجمة عن عملية الادخال .

الاعلان عن البرنامج الجزئي

يتم تعريف البرنامج الجزئي قبل البدء بتعريف المقاطع (Segments) وعند تعريف البرنامج الجزئي يجب مراعاة ما يلي

- يعطى البرنامج الجزئي اسما كما يلي

MACRONAME MACRO

- تُحدد التعليمات المراد استخدامها في البرنامج الجزئي
- انهاء البرنامج الجزئي بالتعليمة ENDM

وبهذا فان البرنامج الجزئي يضم

- جزء التعريف والذي يتضمن اسم البرنامج الجزئي اضافة الى MARCO
- التعليمات والتي تسمى بجسم البرنامج
- نهاية البرنامج الجزئي

والشكل التالي يوضح مكونات البرنامج الجزئي

```
ZZZ     MACRO          ; MACRO DEFINTION
        SET            ;  BODY
        OF             ; OF
        INSTRUCTION    ; MACRO
        ENDM           ;  MACRO END
```

يتم استدعاء البرنامج الجزئي وذلك بذكر اسمه داخل البرنامج الرئيسي والمثال التالي يوضح كيفية الاعلان عن البرنامج الجزئي واستدعائه داخل البرنامج الرئيسي حيث استخدم برنامجان جزئيان الاول (INIT1) لتهيئته المسجلات اللازمة والثاني (DISPM) لعرض رسالة على الشاشة .

```
                                   ; MACRO WITHOUT PARAMETERS
                                   ;------------------------
                                   INIT1    MACRO
                                                      ASSUME CS:CSEG,DS:DSEG,SS:SSEG,
                                   ES:DSEG
                                                      PUSH DS
                                                      SUB AX,AX
                                                      PUSH AX
                                                      MOV AX,DSEG
                                                      MOV DS,AX
                                                      MOV ES,AX
                                                      ENDM
                                   ;----------------------------
                                   DISPM    MACRO
                                                      MOV AH,40H
                                                      MOV BX,01
                                                      MOV CX,12
                                                      LEA DX,MESS
                                                      INT 21H
                                                      ENDM
                                   ;-----------------------------
0000                               SSEG               SEGMENT PARA STACK 'STACK'
0000   0020[                                          DW 32 DUP(?)
           ????
                          ]

0040                               SSEG               ENDS
                                   ;------------------------------
0000                               DSEG               SEGMENT PARA 'DATA'
0000   4D 41 43 52 4F 20           MESS               DB 'MACRO EXAMPLE', 13
       45 58 41 4D 50 4C
       45 0D
000E                               DSEG               ENDS
                                   ;--------------------------------
0000                               CSEG               SEGMENT PARA 'CODE'
0000                               BEGIN    PROC FAR
                                                      INIT1
0000   1E                     1                       PUSH DS
0001   2B C0                  1                       SUB AX,AX
0003   50                     1                       PUSH AX
0004   B8 ---- R              1                       MOV AX,DSEG
0007   8E D8                  1                       MOV DS,AX
0009   8E C0                  1                       MOV ES,AX
```

```
                                                  DISPM
000B   B4 40                  1                   MOV  AH,40H
000D   BB 0001                1                   MOV  BX,01
0010   B9 000C                1                   MOV  CX,12
0013   8D 16 0000 R           1                   LEA  DX,MESS
0017   CD 21                  1                   INT  21H
0019                            BEGIN    ENDP
0019                            CSEG              ENDS
                                                  END      BEGIN
```

لاحظ انه تم استدعاء البرامج الجزئية بذكر اسمها داخل البرنامج الرئيسي كما وان تعليمات البرنامج الجزئي تم عرضها بعد طباعة البرنامج (LST) وان التعليمات الخاصة بالبرنامج الجزئي قد ابتدأت بالرقم (1) والذي بدوره يدل على ان هذه التعليمات تابعة للبرنامج الجزئي .

وقد يرتبط البرنامج الجزئي بمتغير او مجموعة من المتغيرات (PARAMETERS) وفي هذه الحالة يتم الاعلان عن هذه المتغيرات في جزء التعريف وعند الاستدعاء يتم ربط اسم البرنامج الجزئي بذكر اسمه مصحوبا بالمتغيرات كما يلي

ZZZ MACRO MA1 , B1 , C1

ويتم استدعاء هذا البرنامج الجزئي بذكر اسمه مصحوبا بالمتغيرات كما يلي

ZZZ C, DM , K

وعند هذا يجب مراعاة الامور التالية :

– عدد المتغيرات في جملة التعريف يجب ان يكون مساويا لعدد المتغيرات في جملة الاستدعاء .

– يجب ان تتطابق المتغيرات في النوع فمثلا المتغير C يطابق MA1 ، المتغير DM يطابق B1 والمتغير K يطابق C1 هذا ويمكن تسمية المتغيرات في جملة الاستدعاء وفي جملة الاعلان بنفس الاسماء .

– تقسم المتغيرات الى قسمين

١ – متغيرات معروفة القيمة في البرنامج الرئيسي حيث يتم تمرير قيمتها الى المتغيرات المناظرة في البرنامج الجزئي .

٢ – متغيرات غير معروفة القيمة يتم احتسابها داخل البرنامج الجزئي وبعدها يتم تمريرها الى المتغيرات المناظرة في البرنامج الرئيسي .

والمثال التالي يوضح كيفية استخدام المتغيرات في البرنامج الجزئي

(DISPM) , (INITI)

```
                                         ; MACRO WITH    PARAMETERS
                                         ;-------------------------
                                         INIT1   MACRO CSEM,DSEM,SSEM,DSEM
                                                         ASSUME CS:CSEM,DS:DSEM,SS:SSEM,
                                         ES:DSEM
                                                         PUSH DS
                                                         SUB AX,AX
                                                         PUSH AX
                                                         MOV AX,DSEM
                                                         MOV DS,AX
                                                         MOV ES,AX
                                                         ENDM
                                         ;-----------------------------
                                         DISPM   MACRO  MESSM
                                                         MOV AH,40H
                                                         MOV BX,01
                                                         MOV CX,12
                                                         LEA DX,MESS
                                                         INT 21H
                                                         ENDM
                                         ;-----------------------------
0000                                     SSEG            SEGMENT PARA STACK 'STACK'
0000   0020[                          .                  DW 32 DUP(?)
           ????
                               ]

0040                                     SSEG            ENDS
                                         ;-----------------------------
0000                                     DSEG            SEGMENT PARA 'DATA'
0000   4D 41 43 52 4F 20                 MESS            DB 'MACRO EXAMPLE', 13
       45 58 41 4D 50 4C
       45 0D
000E                                     DSEG            ENDS
                                         ;-------------------------------
0000                                     CSEG            SEGMENT PARA 'CODE'
0000                                     BEGIN   PROC FAR
                                                         INIT1 CSEG,DSEG,SSEG,DSEG
0000   1E                           1                    PUSH DS
0001   2B C0                        1                    SUB AX,AX
0003   50                           1                    PUSH AX
0004   B8 ---- R                    1                    MOV AX,DSEG
0007   8E D8                        1                    MOV DS,AX
0009   8E C0                        1                    MOV ES,AX
                                                         DISPM  MESS
000B   B4 40                        1                    MOV AH,40H
000D   BB 0001                      1                    MOV BX,01
0010   B9 000C                      1                    MOV CX,12
0013   8D 16 0000 R                 1                    LEA DX,MESS
0017   CD 21                        1                    INT 21H
0019                                     BEGIN   ENDP
0019                                     CSEG            ENDS
                                                         END     BEGIN
```

التعليمات LALL , XALL , SALL

لاحظنا في الامثلة السابقة انه عند طباعة البرنامج فان تعليمات البرنامج الجزئي قد تكررت في البرنامج الرئيسي (التعليمات المسبوقة بالرقم 1) وتمتلك لغة التجميع بعض التعليمات الخاصة بعرض او عدم عرض تعليمات البرنامج الجزئي عند استدعائه في البرنامج الرئيسي ومن هذه التعليمات :

١ - LALL (LIST ALL) حيث تستخدم هذه التعليمة لعرض كافة تعليمات البرنامج الجزئي عن استدعائه في البرنامج على ان تسبق هذه التعليمة تعليمة الاستدعاء .

٢ - SALL (SUPPRESS ALL) ويؤدي استخدام هذه التعليمة الى تلافي تكرار تعليمات البرنامج الجزئي عند استدعائه داخل البرنامج الرئيسي .

٣ - XALL حيث تستخدم هذه التعليمة لعرض التعلميات التنفيذية فقط (التي تولد شيفرة هدفية object code) اما التعليمات الغير منفذة فلا يتم عرضها (مثل الملاحظة) .

وتجدر الاشارة هنا الى ان استخدام احدى هذه التعليمات يبقي مفعولها قائما على كافة البرامج الجزئية المستخدمة ما لم تستخدم تعليمة اخرى والمثال التالي يوضح كيفية استخدام هذه التعليمات واثرها على عرض تعليمات البرامج الجزئية .

```
                            ;    USING OF LALL,XALL,SALL
                            ;    ----------------------
                            MSG      MACRO    MESSAGE
                            ;MACRO TO DISPLAY ANY MESSAGE
                            ;----------------------------
                                              MOV AH,09
                                              LEA DX,MESSAGE
                                              INT 21H
                                     ENDM
                            ;-----------------------------
0000                        STACK    SEGMENT  PARA STACK 'STACK'
0000   0020[                                  DW 32 DUP(?)
           ????

0040                        STACK    ENDS
                            ;-----------------------------
0000                        DATASG   SEGMENT  PARA 'DATA'
0000   48 45 4C 4C 4F 21    MESS1    DB       'HELLO!', '$'
       24
0007   47 4F 4F 44 20 42    MESS2    DB       'GOOD BY', '$'
       59 24
000F   54 48 41 4E 4B 53    MESS3    DB       'THANKS', '$'
       24
0016                        DATASG   ENDS
```

```
                                         ;--------------------------------
0000                                     CODESG  SEGMENT PARA 'CODE'
0000                                     BEGIN   PROC FAR
                                                         ASSUME CS:CODESG,DS:DATASG,SS:S
                                         TACK,ES:DATASG
0000  1E                                                 PUSH DS
0001  2B C0                                              SUB AX,AX
0003  50                                                 PUSH AX
0004  B8 ---- R                                          MOV AX,DATASG
0007  8E D8                                              MOV DS,AX
0009  8E C0                                              MOV ES,AX
                                         ; MACRO WITH LIST ALL
                                         ;--------------------
                                                         .LALL
                                                         MSG  MESS1
                                     1   ;MACRO TO DISPLAY ANY MESSAGE
                                     1   ;----------------------------
000B  B4 09                          1                   MOV AH,09
000D  8D 16 0000 R                   1                   LEA DX,MESS1
0011  CD 21                          1                   INT 21H
                                         ; MACRO WITH SUPPRESS ALL SALL
                                         ;-----------------------------
                                                         .SALL
                                                         MSG  MESS2
                                         ;  MACRO WITH XALL
                                         ;------------------------------
                                                         .XALL
                                                         MSG  MESS3
001B  B4 09                          1                   MOV AH,09
001D  8D 16 000F R                   1                   LEA DX,MESS3
0021  CD 21                          1                   INT 21H
0023                                     BEGIN   ENDP
0023                                     CODESG  ENDS
                                                         END     BEGIN
```

تداخل البرامج الجزئية

استعرضنا سابقا عملية استدعاء البرنامج الجزئي من خلال البرنامج الرئيسي وقد يتم استدعاء البرنامج الجزئي من داخل برنامج جزئي وفي هذه الحالة يمكن اعتبار البرنامج الجزئي المستدعي لبرنامج جزئي آخر بالبرنامج الرئيسي .

فمثلا البرنامج الجزئي التالي يخزن شيفرة تعليمة نظم التشغيل اللازمة للاعتراض (21H) في المسجل AH

```
INT21  MACRO   FUNC1
       MOV  AH , FUNC1
       INT  21H
       ENDM
```

لنفرض اننا نريد عرض رمز معين على الشاشة باستخدام برنامج جزئي لذا فإن هذا البرنامج يمكن ان يتضمن

```
CHARD    MACRO  CHAR
         MOV    AH , 02
         MOV    DL,  CHAR
         INT    21H
         ENDM
```

ولتحقيق نفس الغرض يمكن استخدام البرنامج الجزئي الاول ضمن البرنامج الجزئي الثاني بحيث يمكن اعتبار البرنامج الثاني رئيسيا بالنسبة للاول

```
CHARD    MACRO  CHAR
         MOV  DL , CHAR
         INT21   02
         ENDM
```

وفي هذه الحالة يجب تعريف كلا البرنامجين قبل البدء بتعريف المقاطع في البرنامج الرئيسي .

التعليمة LOCAL

في بعض الاحيان قد يستدعى البرنامج الجزئي اكثر من مرة في البرنامج الرئيسي وفي هذه الحالة سوف يظهر خطأ تكرار الاسماء (Duplicate names) ولتلافي هذا لا بد من استخدام LOCAL لتعريف عناصر البيانات وعلامات (Labels) التعليمات لاعطاء كل عنصر او علامة اسما فريدا بعد كل عملية استدعاء .

ويتم تعريف عناصر البيانات والعلامات المستخدمة في البرنامج الجزئي مباشرة بعد تعريفه كما يلي

LOCAL dummy1 , dummy2 , ...

والمثال التالي يوضح كيفية استخدام التعليمة LOCAL

```
; USING LOCAL IN MACRO
;----------------------
ZZZ              MACRO A1,B1,C1
                 LOCAL LL1
                 LOCAL LL2
                 MOV AX,A1
                 MOV BX,B1
                 SUB CX,CX
```

```
                   LL1:
                                      CMP  AX,BX
                                      JB   LL2
                                      SUB  AX,BX
                                      INC  CX
                                      JMP  LL1
                   LL2:
                                      MOV  C1,CX
                                      ENDM
                   ;---------------------------
0000               CODS               SEGMENT PARA 'CODE'
                                      ASSUME CS:CODS,DS:CODS,SS:CODS,
                   ES:CODS
0100                                  ORG   100H
0100   EB 06       BEGIN:   JMP   SHORT MAIN
                   ;---------------------------
0102   00C8        DIV1               DW   200
0104   0023        DSR1               DW   35
0106   0000        QTT1               DW   ?
                   ;---------------------------
0108               MAIN               PROC NEAR
                                      .SALL
                                      ZZZ  DIV1,DSR1,QTT1
011E   C3                             RET
011F               MAIN               ENDP
011F               CODS               ENDS
                                      END       BEGIN
```

عند استدعاء البرنامج الجزئي ZZZ للمرة الاولى فان العلامة (LL1) سوف تعطى الرقم 0000؟؟ والعلامة (LL2) سوف تعطى الرقم (0001؟؟) في البرنامج الرئيسي .

وعند الاستدعاء الثاني فانها سوف تعطى الارقام (0002؟؟) و (0003؟؟)

ملاحظة : يمكن ملاحظة هذا عند استبدال التعليمة SALL بـ LALL .

مكتبة البرامج الجزئية

MACRO LIBRARY

من ما سبق نخلص الى القول ان البرامج الجزئية تؤدي الى عدم تكرار التعليمات التي تؤدي هدف معين داخل البرنامج الرئيسي وذلك بجمع هذه التعليمات في برنامج جزئي .

ولكن قد يتكرر استخدام البرنامج الجزئي الواحد في اكثر من برنامج وبدلا من عملية التكرار هذه لا بد من حفظ البرامج الجزئية في مكتبه لغاية الرجوع اليها وفي اي برنامج قد يحتاج اليها .

لذا فبدلا من تعريف البرنامج الجزئي في البرنامج الرئيسي يمكن الرجوع الى مكتبة البرامج الجزئية (MACRO . LIB) وذلك باستخدام التعليمة

INCLUDE C : MACRO . LIB

(على فرض ان المكتبة مخزنة على الوحدة C) .

حيث تسد هذه التعليمة مكان تعريف البرامج الجزئية المطلوبه ويتم استدعاء البرنامج الجزئي كما اشرنا سابقا بذكر اسمه (على ان يكون ضمن المكتبة) . وفي هذه الحالة وعند طباعة البرنامج الرئيسي فان تعليمات البرنامج الجزئي هي الاخرى سوف تطبع وللتخلص من هذا ولتوفير الوقت وورق الطباعة يمكن استخدام جملة الالحاق (Include) بالشكل التالي

```
IF1
    INCLUDE C : MACRO . LIB
ENDIF
```

يؤدي استخدام تعليمة (Include) الى الحاق المكتبة (كافة البرامج الجزئية المتوفرة في المكتبة) بالبرنامج الرئيسي حيث ترتبط به ولكن قد لا يحتاج البرنامج الرئيسي الى كافة البرامج في المكتبة لذا لا بد من فك ارتباط البرنامج بالبرامج الجزئية الغير مطلوبة (يتم الغاؤها من البرنامج الرئيسي ولكنها تبقى في المكتبة) وتستخدم لهذه الغاية التعليمة PURGE .

لنفرض ان المكتبة تحتوي على البرامج الجزئية MAC1 , MAC2 , MAC3 وان البرنامج الرئيسي يحتاج فقط الى MAC1 ولحذف MAC3 , MAC2 من البرنامج الرئيسي يمكن تطبيق الآتي :

```
IF1
    INCLUDE   MACRO . LIB
ENDIF
PURGE   MAC2 , MAC3
```

التكرار المنتهي بـ ENDM

يستخدم هذا التكرار لتكرار مجموعة من التعليمات داخل البرنامج الرئيسي او داخل البرنامج الجزئي على ان ينتهي بالامر ENDM والذي يعني نهاية التكرار وليس البرنامج الجزئي ولانهاء هذا البرنامج لا بد من استخدام امر ENDM آخر ومن اهم التكرارات المستخدمة والمنتهية بالامر ENDM نورد ما يلي

- التكرار REPT

يستخدم هذا التكرار لتكرار مجموعة من التعليمات منهية بالامر ENDM عدد محدد من المرات

والمثال التالي يوضح مفهوم هذا التكرار

```
K = 0
REPT 10
S = S + 1
DW S
ENDM
```

حيث يتم بواسطة هذا التكرار توليد ١٠ جمل (... , DW1 , DW2) والتعليمات التالية تكرر تنفيذ MOVSB خمس مرات

```
REPT 5
MOVSB
ENDM
```

- التكرار IRP (Indefinite Repeat)

يأخذ هذا التكرار الشكل التالي

```
IRP dummy , < arguments >
```

فمثلا توليد الجمل 10 , DB 5 , DB 12 , DB 7 DB يمكن باستخدام التكرار التالي

```
IRP K < 10, 5, 12, 7 >
DB K
ENDM
```

- التكرار IRPC (Indefineite Repeat character)

يأخذ هذا التكرار الشكل التالي

```
IRPC dummy , string
```

وكمثال على هذا نورد

```
IRPC M , 1238
DW M
ENDM
```

حيث يتم توليد DW1 , DW2 , DW3 , DW8 بهذا التكرار

خاصية الضم في البرنامج الجزئي

CONCATE NATION(&)

يستخدم الرمز & لاخبار المترجم بضم نص او رمز معين ويمكن بيان هذا من خلال استخدام MOVSW , MOVSB في البرنامج الجزئي التالي

```
MOVING  MACRO  CH
        REP  MOVS & CH
        ENDM
```

فاذا كانت قيمة CH مساوية لـ B فسوف تنفذ التعليمة REP MOVSB اما اذا كانت مساوية لـ W فسوف تنفذ التعليمة REP MOVSW

التعليمات الشرطية

CONDITIONAL OPERATIONS

تتوفر في لغة التجميع امكانية استخدام الجمل الشرطية في البرنامج الجزئي او في البرنامج الرئيسي وتأخذ جملة IF الشكل التالي

```
IF xx  (condition)
.
.
EISE  (optional)
.
.
ENDIF  (end of IF)
```

فاذا كانت قيمة الشرط صائبة (true) فسوف تنفذ الجمل لغاية EISE اما اذا كانت خاطئة فسوف تنفذ الجمل التي تلي جملة ELSE فقط .

وتأخذ جملة IF عدة أشكال :

- IF expression

يتم التحقق من قيمة التعبير فاذا كان لا يساوي الصفر يتم تنفيذ الشرط (يعتبر الشرط صائبا)

فمثلا

```
IF A
EXITM
ENDIF
```

فاذا كانت A لا تساوي الصفر يتم الخروج من البرنامج الجزئي وذلك بتنفيذ أمر الخروج EXITM

- IFE expression

يشبه هذا الشكل الشكل السابق ولكن الشرط يعتبر صائبا اذا كان التعبير مساويا للصفر .

- IF1 (no expression)

يعتبر الحدث صائبا اذا كانت عملية التجميع في المرحلة الاولى (PASS 1) وهي المرحلة التي يتم فيها توليد جداول الرموز الخاصة .

- IF2 (no expression)

يعتبر الحدث صائبا اذا كانت عملية التجميع في المرحلة الثانية (PASS 2) وهي المرحلة التي تستخدم فيها الجداول المبنية في المرحلة الاولى لربط التعليمات .

- IFDEF symbol

اذا كان الرمز المرتبط بجملة IF هذه معرفا فان الحدث صائب

- IFNDEF symbol

اذا كان الرمز (symbol) غير معرف فان الحدث صائب

- IFB <argument >

تكون قيمة الحدث صائبة اذا كانت قيمة المتغير (argument) مساوية للفراغ (Blank)

IFNB <argument >

تكون قيمة الحدث صائبة اذا كانت قيمة المتغير لا تساوي الفراغ .

- IFIND < arg 1 > , <arg 2 >

يكون الحدث صائبا اذا تشابهت قيم arg 2 , arg 1

- IFDIF <arg 1 > , <arg 2 >

يكون الحدث صائبا اذا اختلفت قيم arg2 , arg1

والامثلة التالية توضح المفاهيم الاساسية لجملة IF

```
IFNBM  MACRO  FUNC , DXA
       MOV  AH , FUNC
```

```
IFNB < DXA >
MOV  DX , OFFSET  DXA ; 1
ENDIF
INT 21H
ENDM
```

فاذا كانت DXA = BLANK فان الجملة (١) سوف لا تنفذ

```
MOVIF  MACRO  CH
       IFIDN  < & CH > , < B >
       REP MOVSB
       EXITM
       ENDIF
       IFIDN < & CH > , < W >
       REP MOVSW
       ELSE
       REP MOVSB
       ENDIF
       ENDM
```

فاذا كانت قيمة الرمز مساوية لـ B فسوف تنفذ التعليمة MOVSB وبعدها يتم الخروج من البرنامج الجزئي .

أما اذا كانت قيمة الرمز W فسوف تنفذ MOVSW ، واذا كانت قيمة الرمز مساوية لاية قيمة اخرى عدا B او W فسوف تنفذ MOVSB والبرنامج التالي يوضح كيفية استخدام بعض اشكال جملة IF في البرنامج الجزئي

```
;   USING IF AND IFNDEF IN MACRO
;-------------------------------
ZZZ                MACRO  A1,B1,C1
                   LOCAL  LL1
                   LOCAL  LL2
                   CNTR=0
         IFNDEF    A1
; A1 NOT DEFINED
                   CNTR=CNTR+1
         ENDIF
         IFNDEF    B1
                   CNTR=CNTR+1
```

```
                  ENDIF
                  IFNDEF  C1
                          CNTR=CNTR+1
                  ENDIF
                  IF      CNTR
                  EXITM
                  ENDIF
                          MOV AX,A1
                          MOV BX,B1
                          SUB CX,CX
              LL1:
                          CMP AX,BX
                          JB LL2
                          SUB AX,BX
                          INC CX
                          JMP LL1
              LL2:
                          MOV C1,CX
                  ENDM
              ;----------------------------
0000          CODS        SEGMENT PARA 'CODE'
                          ASSUME CS:CODS,DS:CODS,SS:CODS,
              ES:CODS
0100                      ORG 100H
0100          BEGIN:
0100   EB 06              JMP SHORT MAIN
0102   00C8   A1          DW 200
0104   0023   B1          DW 35
0106   0000   C1          DW ?
0108          MAIN        PROC NEAR
                          .SALL
                          ZZZ A1,B1,C1
011E   C3                 RET
011F          MAIN        ENDP
011F          CODS        ENDS
                          END      BEGIN
```

الوحدة الثانية عشرة

معالجة الملفات

- استحداث الملف
- القراءة التتابعية للملف
- المعالجة العشوائية
- المعالجة العشوائية للكتل
- العمليات المتعلقة بالقرص المغناطيسي
- الاقراص الممغنطة وبرمجيات الادخال والاخراج

معالجة الملفات

FILE PROCESSING

يُعرف الملف على انه مجموعة من السجلات المتجانسة بحيث يتضمن السجل مجموعة من الحقول المختلفة ، وقبل البدء في الحديث عن معالجة الملفات لا بد من القاء الضوء على واسطة التخزين المستخدمة لحفظ الملفات (الاقراص المغناطيسية) .

تخزن الملفات عادة على الاقراص المغناطيسية المرنة (Floppy disks) او الاقراص المغناطيسية الصلبة (Hard disks) وتتنوع هذه الاقراص في سعتها ولكنها تتشابه في عملية تقسيمها الى مقاطع (sectors) و مسارات (tracks) والجدول التالي يبين اهم انواع هذه الاقراص وسعة كل نوع منها

	Version	Tracks /side	secters /track	Bytes /sector	TOTAL 2 sides
FLOPPY	5 1/4"	40	9	512	368640
disk	3 1/2 "	80	9	512	737280
	3 1/2" HD	80	15	512	1,228800
Hard	10 Mbyte	306	17	512	10653696
disk	20 Mbyte	614	17	512	21377024

ترقم الأوجه (sides) والمسارات بدأ من الرقم (0) بينما ترقم المقاطع بدأ من الرقم (1)

ولحساب الحجم المتبقي (الغير محجوز) من القرص المغناطيسي تقوم نظم التشغيل (DOS) بحجز عدد من المقاطع وذلك لاغراض خاصة فمثلا في القرص المغناطيسي ثنائي الوجه تتم عمليات التخصيص التالية

- side 0 , track 0 , sector 1 Boot record
- side 0 , track 0 , sectors 2-3 FAT *
- side 0 , track 0 , sectors 4-7 Directory
- side 1 , track 0 , sectors 1-3 Directory
- side 1 , track 0 , sectors 4 and on Data file

* FAT File allocation table جدول حجز الملف

يتم تخزين السجلات في المسارات بشكل متتابع بحيث تخزن البيانات في مسار ما في الوجه الاول ثم في المسار المناظر في الوجه الآخر وذلك لتقليل حركة رؤوس القراءة والكتابة وزيادة سرعة الوصول .

اما الفهرس (Directory) فيتم تخصيصه لكل ملف بحيث يحتوي على معلومات هامة كإسم الملف ، تاريخ الانشاء الحجم والموقع ويتألف الفهرس من ٣٢ بايتا تتضمن المعلومات التالية

البايت	الهدف
0 - 7	اسم الملف كما هو في برنامج انشاء الملف ويخصص أول بايت للدلالة على حالة الملف (00H تشير الى ان الملف لم يستخدم بعد ، E5H تشير الى ان الملف قد شطب، 2EH تشيرالى فهرس فرعي).
8 - 10	نوع الملف (Extension)
11	نوع الملف من حيث المعالجة (00H ملف عادي ، 01H ملف قراءة فقط ، 02H ملف مخفي (hidden) ،04H ملف نظم تشغيل ، 08H علامة المنطقة (Volume Label) ، 10H فهرس فرعي ، 20H ملف ارشيف خاص للقرص الصلب) .
12 - 21	محجوزة لنظم التشغيل
22 - 23	زمن انشاء الملف او آخر تعديل تم على الملف .
24 - 25	تاريخ الانشاء او تاريخ آخر تعديل .
26 - 27	موقع الملف
28 - 31	حجم الملف بالبايت .

اما جدول حجز الملف (FAT) فيبدأ بالقاطع ٢ ويحتوي هذا الجدول على مجموعة من المداخل بحيث يشير البايت الاول في هذا الجدول الى نوع وحدة التخزين اما البايت الثاني والثالث فتضم المقاطع الاول والاخير والموجود فيها الملف وبعد هذا يضم الجدول مجموعة اخرى من البايت تحدد المقاطع المستخدمة للملف بحيث يحدد في احد هذه القواطع موقع الفهرس اما وجود البيانات فيحدد برقم القاطع بحيث يشير القاطع دائما الى القاطع التالي ووجود الرقم (000) يشير الى نهاية البيانات .

استحداث الملف
FILE CREATION

تتم عملية معالجة الملفات وذلك من خلال انشاء كتلة التحكم بالملف (File control Block FCB) والذي يتضمن ٣٦ بايتا يتم التحكم بها من خلال المبرمج - 32 , 15 - 0) (36 ومن خلال نظم التشغيل (31 - 16) وتتضمن هذه البايتات كافة المعلومات اللازمة لمعالجة الملف وفيما يلي جدول يبين المعلومات المخزنة في كل بايت من كتله التحكم هذه.

البايت	الاستخدام
0	نوع وحدة التخزين (01 الوحدة A ، 02 الوحدة B وهكذا) .
1 -8	اسم الملف
9 - 11	ملحق اسم الملف (Extension)
12 - 13	رقم الكتلة والتي تحتوي على ١٢٨ سجلاً ويتم ترقيم الكتل نسبة الى بداية الملف ابتداء من الصفر حيث تستخدم هذه الكتل اثناء عمليات القراءة والكتابة ، ويتم تصفير هذا الحقل عند فتح الملفات .
14 - 15	طول السجل المنطقي وعند فتح الملف يتم اعطاء هذا الطول القيمة ١٢٨ وبعد عملية الفتح وقبل اجراء عملية الكتابة يمكن تغير هذا الطول الى القيمة المطلوبة .
16 - 19	حجم الملف والذي يتم اخذه من الفهرس بواسطة نظم التشغيل حيث يثبت في هذا الحقل وتقوم نظم التشغيل بتعديله اذا لزم الامر .
20 - 21	تاريخ انشاء الملف والذي يؤخذ هو الآخر من الفهرس ويعدل من قبل نظم التشغيل .
22 - 31	محجوزة لنظم التشغيل
32	رقم السجل الحالي (127 - 0) في الكتلة الحالية (الحقل 13 - 12)
33 - 36	رقم السجل النسبي (Relative RENCO)

ولاستحداث الملف يخزن عنوان FCB لهذا الملف في السجل DX وتنفذ وظيفة الحجز (16H) حيث تتم عملية الاستحداث حسب الخطوات التالية

```
MOV   AH , 16H        ; Create
LEA   DX , FCBname    ; Address of FCB
INT   21H             ; Call DOS
```

حيث تقوم نظم التشغيل بأخذ اسم الملف وتبحث عنه في الفهرس فإذا كان موجودا يتم استخدام نفس المكان في الفهرس اما اذا لم يكن موجودا فيتم ايجاد مدخل جديد في الفهرس لهذا الملف . وتقوم بعد هذا التعليمة OPEN بوضع صفر في حقل حجم الملف وتقوم بالتأكد من وجود مكان خالي وذلك بفحص محتوى المسجل AL فاذا كان يحتوي على (00) فهذا يعني وجود مكان اما اذا كان محتواه (FF) فان هذا يعني عدم توفر مكان للملف .

كما وتقوم التعليمة OPEN بتصفير رقم الكتلة الحالي في FCB وتقوم ايضا بتخزين الرقم ١٢٨ في حقل طول السجل .

وبعد فتح الملف تحضر منطقة تبادل البيانات (Dara tranfer area DTA) والخاصة بتسجيل السجلات وقبل البدء بعملية الكتابة (Write) لا بد من تحديد عنوان (DTA) ويتم هذا حسب الخطوات التالية :

```
MOV    AH , 1AH         ; set addres
LEA    DX , DTAname     ; OF DTA
INT    21H              ; Call DOS
```

فاذا كان البرنامج يتعامل مع ملف واحد فان عملية تحديد DTA تتم مرة واحدة اما اذا كان هناك اكثر من ملف فلا بد من تحديد DTA قبل اجراء كل عملية قراءة او كتابة .

اما عملية الكتابة فتم حسب الخطوات التالية

```
MOV    AH , I5H         ; sequentially
LEA    DX , FCBname     ; write a record
INT    21H              ; Call DOS
```

تستخدم تعليمة الكتابة المعلومات الموجودة في FCB وعنوان DTA وتقوم بتسجيل السجل في القاطع اذا كان طول السجل مساويا لطول القاطع وإلا تتم تعبئة السجل في منطقة تخزين مؤقتة (buffer) حتى يصبح الحجم مساويا لحجم القاطع عندها تُنقل السجلات الى القاطع في القرص المغناطيسي وعند نجاح عملية القراءة تقوم نظم التشغيل بالخطوات التالية : -

- تعبئة حقل حجم الملف في FCB او زيادته بمقدار طول السجل عند اضافة السجل .
- زيادة رقم السجل الحالي وعند خروج الرقم عن ١٢٨ يتم تصفيره بعد اضافة ١ الى رقم الكتلة في FCB .
- ارجاع شيفرة الكتابة الى المسجل AL وذلك لفحص عملية الكتابة فاذا كانت

الشيفرة (00) فان عملية الكتابة ناجحة ، (10) القرص ممتلىء ، (02) لا يوجد هناك حيز لسجل واحد في DTA .

وبعد اجراء عمليات الكتابة (تسجيل السجلات) وقبل انهاء عملية استحداث الملف لا بد من اغلاق (CLOSE) هذا الملف حيث تتم عملية الاغلاق حسب الخطوات التالية

```
MOV   AH , 10H          ; File
LEA   DX , FCBname      ; Closing
INT   21H               ; Call DOS
```

وفيما يلي نستعرض مثالا لاستحداث ملف يتضمن مجموعة من الاسماء يتم ادخالها عن طريق لوحة المفاتيح .

تتضمن FCB المداخل التالية :

FCBDRIV	استحداث الملف يتم على وحدة الاقراص رقم ٣ (C)
FCBNAME	اسم الملف NAMEFILE
FCBEXT	ملحق الاسم DTA
FCBBLK	رقم الكتلة الحالي يبدأ من الصفر
FCBRCSZ	طول السجل (في البداية يتم اعتباره مساويا لـ ١٢٨) .
FCBQRC	رقم السجل الحالي (في البداية قيمة هذا الرقم صفر) .

```
OPENM          MACRO
                        LOCAL  ZZZ
                   LOCAL  EEE
                   MOV  AH,16H
                   LEA  DX,FCBREC
                   INT  21H
                 CMP  AL,00
; AVIALABLE SPACE ? NO ERROR
               JNZ  ZZZ
                   MOV  FCBRCSZ,RECLEN
                   LEA  DX,NAMEDTA
;  SET ADDRESS OF DTA
                   MOV  AH,1AH
                   INT  21H
               JMP  EEE
ZZZ:           LEA  DX,OPNMSG
                   ERRM
;  MACRO  ERROR
EEE:
               ENDM
ERRM               MACRO
;-----------------------------
;  MACRO  TO  DETECT  ERRORS
;  --------------------
```

```
                MOV AH,09
                INT 21H
                MOV ERRCDE,01
                ENDM
DISPM        MACRO
                LOCAL LM10
                LOCAL LM20
; MACRO TO SCROLL AND SET CURSOR
; ------------------------------
                MOV AH,09
                LEA DX,CRLF
                INT 21H
                CMP ROW,18
                JAE LM20
                INC ROW
             JMP LM10
LM20:
                MOV AX,0601H
                SCRM
                CURSM
LM10:
                ENDM
SCRM         MACRO
; MACRO TO SCROLL SCREEN
; ----------------------
                MOV BH,1EH
                MOV CX,0000
                MOV DX,184FH
                INT 10H
                ENDM



CURSM        MACRO
;  MACRO TO SET CURSOR
;  -----------------
                MOV AH,02
                MOV BH,00
                MOV DL,00
                MOV DH,ROW
                INT 10H
                ENDM
CLSEM        MACRO
;  MACRO TO CLOSE FILE
;  -----------------
                MOV NAMEDTA,1AH
                WRITM
                MOV AH,10H
                LEA DX,FCBREC
                INT 21H
                ENDM
WRITM        MACRO
                LOCAL W20
;  MACRO TO WRITE DISK RECORD
;  ------------------------
```

```
                    MOV AH,15H
                    LEA DX,FCBREC
                    INT 21H
                    CMP AL,00
                    JZ W20
                    LEA DX,WRTMSG
                    ERRM
                    MOV NAMELEN,00
W20:
                    ENDM
CLSEM           MACRO
                    MOV NAMEDTA,1AH
                    WRITM
                    MOV AH,10H
                    LEA DX,FCBREC
                    INT 21H
                    ENDM
PROCM           MACRO
                LOCAL P10
                    LOCAL P20
                    MOV AH,09
                    LEA DX,PROMPT
                    INT 21H
                    MOV AH,0AH
                    LEA DX,NAMEPAR
                    INT 21H
                    DISPM
                    CMP NAMELEN,00
                    JNE P20
                JMP P10
P20:
                    MOV BH,00

                                        MOV BL,NAMELEN
                                        MOV NAMEDTA[BX],' '
                                        WRITM
                                        CLD
                                    LEA DI,NAMEDTA
                                        MOV CX,RECLEN/2
                                        MOV AX,2020H
                                        REP STOSW
                            P10:
                                        ENDM
                            ;-------------------------------------------
                            ; PROGRAM TO CREAT DISK FILE USING FCB
0000                        SSEG              SEGMENT PARA STACK 'STACK'
0000   0050[                                  DW 80 DUP(?)
          ????
                     ]

00A0                        SSEG              ENDS
                            ;-------------------------------------------
0000                        DSEG              SEGMENT PARA 'DATA'
= 0020                      RECLEN  EQU 32
0000                        NAMEPAR LABEL BYTE
0000   20                   MAXNLEN DB RECLEN
0001   00                   NAMELEN DB ?
0002   0020[                NAMEDTA DB RECLEN DUP(' ')
          20
```

```
                                            ]
0022                                  FCBREC   LABEL BYTE
0022   03                             FCBDRIV  DB 03
0023   4E 41 4D 45 46 49              FCBNAME  DB 'NAMEFILE'
       4C 45
002B   44 41 54                       FCBEXT   DB 'DAT'
002E   0000                           FCBBLK   DW 0000
0030   0000                           FCBRCSZ  DW ?
0032   00000000                       FCBFLSZ  DD ?
0036   0000                                             DW ?
0038   000000000000000000                               DT ?
       00
0042   00                             FCBSORC  DB 00
0043   00000000                                         DD ?
0047   0D 0A 24                       CRLF              DB 13, 10, '$'
004A   00                             ERRCDE   DB 00
004B   4E 41 4D 45 3F 20              PROMPT   DB 'NAME? ','$'
       24
0052   01                             ROW               DB 01
0053   2A 2A 2A 4F 50 45              OPNMSG   DB '***OPEN ERROR***', '$'
       4E 20 45 52 52 4F
       52 2A 2A 2A 24
0064   2A 2A 2A 57 52 49              WRTMSG   DB '***WRITE ERROR***', '$'
       54 45 20 45 52 52
       4F 52 2A 2A 2A 24
0076                                  CNTR              LABEL BYTE
0076                                  DSEG              ENDS


                                  ;----------------------------------------
0000                              CSEG              SEGMENT PARA 'CODE'
0000                              BEGIN     PROC FAR
                                                    ASSUME CS:CSEG,DS:DSEG,SS:SSEG,
                                  ES:DSEG
0000   1E                                           PUSH DS
0001   2B C0                                        SUB AX,AX
0003   50                                           PUSH AX
0004   B8 ---- R                                    MOV AX,DSEG
0007   8E D8                                        MOV DS,AX
0009   8E C0                                        MOV ES,AX
000B   B8 0600                                      MOV AX,0600H
                                                    .LALL
                                                    SCRM
                             1    ; MACRO TO SCROLL SCREEN
                             1    ; ---------------------
000E   B7 1E                 1                      MOV BH,1EH
0010   B9 0000               1                      MOV CX,0000
0013   BA 184F               1                      MOV DX,184FH
0016   CD 10                 1                      INT 10H
                                                    CURSM
                             1    ;  MACRO TO SET CURSOR
                             1    ;  -----------------
0018   B4 02                 1                      MOV AH,02
001A   B7 00                 1                      MOV BH,00
001C   B2 00                 1                      MOV DL,00
001E   8A 36 0052 R          1                      MOV DH,ROW
0022   CD 10                 1                      INT 10H
                                                    OPENM
0024   B4 16                 1                      MOV AH,16H
0026   8D 16 0022 R          1                      LEA DX,FCBREC
002A   CD 21                 1                      INT 21H
002C   3C 00                 1                  CMP AL,00
```

```
                                1   ; AVIALABLE SPACE ? NO ERROR
002E  75 11                     1               JNZ ??0000
0030  C7 06 0030 R 0020         1                   MOV FCBRCSZ,RECLEN
0036  8D 16 0002 R              1                   LEA DX,NAMEDTA
                                1   ;  SET ADDRESS OF DTA
003A  B4 1A                     1                   MOV AH,1AH
003C  CD 21                     1                   INT 21H
003E  EB 0E 90                  1               JMP ??0001
0041  8D 16 0053 R              1   ??0000:         LEA DX,OPNMSG
                                1                   ERRM
                                2   ;-----------------------------
                                2   ;  MACRO TO DETECT ERRORS
                                2   ;  --------------------
0045  B4 09                     2                   MOV AH,09
0047  CD 21                     2                   INT 21H
0049  C6 06 004A R 01           2                   MOV ERRCDE,01
                                1   ;  MACRO ERROR
004E                            1   ??0001:
004E  80 3E 004A R 00                               CMP ERRCDE,00
0053  74 01                                         JZ LOOP1
0055  CB                                            RET


0056                                LOOP1:
                                                    PROCM
0056  B4 09                     1                   MOV AH,09
0058  8D 16 004B R              1                   LEA DX,PROMPT
005C  CD 21                     1                   INT 21H
005E  B4 0A                     1                   MOV AH,0AH
0060  8D 16 0000 R              1                   LEA DX,NAMEPAR
0064  CD 21                     1                   INT 21H
                                1                   DISPM
                                2   ; MACRO TO SCROLL AND SET CURSOR
                                2   ; ---------------------------
0066  B4 09                     2                   MOV AH,09
0068  8D 16 0047 R              2                   LEA DX,CRLF
006C  CD 21                     2                   INT 21H
006E  80 3E 0052 R 12           2                   CMP ROW,18
0073  73 07                     2                   JAE ??0005
0075  FE 06 0052 R              2                   INC ROW
0079  EB 1A 90                  2               JMP ??0004
007C                            2   ??0005:
007C  B8 0601                   2                   MOV AX,0601H
                                2                   SCRM
                                3   ; MACRO TO SCROLL SCREEN
                                3   ; --------------------
007F  B7 1E                     3                   MOV BH,1EH
0081  B9 0000                   3                   MOV CX,0000
0084  BA 184F                   3                   MOV DX,184FH
0087  CD 10                     3                   INT 10H
                                2                   CURSM
                                3   ;  MACRO TO SET CURSOR
                                3   ;  ---------------
0089  B4 02                     3                   MOV AH,02
008B  B7 00                     3                   MOV BH,00
008D  B2 00                     3                   MOV DL,00
008F  8A 36 0052 R              3                   MOV DH,ROW
0093  CD 10                     3                   INT 10H
0095                            2   ??0004:
0095  80 3E 0001 R 00           1                   CMP NAMELEN,00
009A  75 03                     1                   JNE ??0003
009C  EB 37 90                  1               JMP ??0002
009F                            1   ??0003:
```

```
009F   B7 00                  1               MOV BH,00
00A1   8A 1E 0001 R           1               MOV BL,NAMELEN
00A5   C6 87 0002 R 20        1               MOV NAMEDTA[BX],'
                              1               WRITM            .
                              2   ;  MACRO TO WRITE DISK RECORD
                              2   ;  -------------------------
00AA   B4 15                  2               MOV AH,15H
00AC   8D 16 0022 R           2               LEA DX,FCBREC
00B0   CD 21                  2               INT 21H
00B2   3C 00                  2               CMP AL,00
00B4   74 12                  2               JZ ??0006
00B6   8D 16 0064 R           2               LEA DX,WRTMSG
                              2               ERRM
                              3   ;------------------------------


                              3   ;  MACRO TO DETECT ERRORS
                              3   ;  ---------------------
00BA   B4 09                  3               MOV AH,09
00BC   CD 21                  3               INT 21H
00BE   C6 06 004A R 01        3               MOV ERRCDE,01
00C3   C6 06 0001 R 00        2               MOV NAMELEN,00
00C8                          2   ??0006:
00C8   FC                     1               CLD
00C9   8D 3E 0002 R           1           LEA DI,NAMEDTA
00CD   B9 0010                1               MOV CX,RECLEN/2
00D0   B8 2020                1               MOV AX,2020H
00D3   F3/ AB                 1               REP STOSW
00D5                          1   ??0002:
                                  ;       CMP NAMELEN,00
                                  ;           JNE      LOOP1
                                              CLSEM
00D5   C6 06 0002 R 1A        1               MOV NAMEDTA,1AH
                              1               WRITM
                              2   ;  MACRO TO WRITE DISK RECORD
                              2   ;  -------------------------
00DA   B4 15                  2               MOV AH,15H
00DC   8D 16 0022 R           2               LEA DX,FCBREC
00E0   CD 21                  2               INT 21H
00E2   3C 00                  2               CMP AL,00
00E4   74 12                  2               JZ ??0007
00E6   8D 16 0064 R           2               LEA DX,WRTMSG
                              2               ERRM
                              3   ;------------------------------
                              3   ;  MACRO TO DETECT ERRORS
                              3   ;  ---------------------
00EA   B4 09                  3               MOV AH,09
00EC   CD 21                  3               INT 21H
00EE   C6 06 004A R 01        3               MOV ERRCDE,01
00F3   C6 06 0001 R 00        2               MOV NAMELEN,00
00F8                          2   ??0007:
00F8   B4 10                  1               MOV AH,10H
00FA   8D 16 0022 R           1               LEA DX,FCBREC
00FE   CD 21                  1               INT 21H
0100   CB                                     RET
0101                              BEGIN   ENDP
0101                              CSEG            ENDS
                                              END BEGIN
```

القراءة التتابعية للملفات
SEQUENTIAL FILE READING

يجب ان يحتوي برنامج قراءة الملف على نفس كتلة التحكم الخاصة بهذا الملف (FCB) والتي استخدمت عند استحداثه وتتم عملية قراءة الملف حسب الخطوات التالية :

١ - فتح الملف (OPEN) حيث تتم هذه العملية وذلك بتنفيذ العمليات التالية:

```
MOV  AH , OFH        ; Open
LEA  DX , FCBname    ; the file
INT  21H             ; Call DOS
```

ونتيجة لتنفيذ عملية الفتح يتم فحص الفهرس للتأكد من وجود اسم الملف وملحق الاسم فاذا لم يكن موجودا يتم تعبئة المسجل AL بالقيمة (FF) اما اذا كان الاسم موجودا فان القيمة (00) سوف تخزن في المسجل AL ثم يتم تحديد حجم الملف ورقم الكتلة (0) ورقم السجل الحالي (0) وطول السجل .

٢ - تحديد عنوان منطقة تبادل البيانات (DTA) وذلك بتنفيذ التعليمات التالية :

```
MOV  AH , 1AH  ; set address
LEA  DX , DTAname  ; of DTA
INT  21H ; Call DOS
```

٣ - قراءة الملف تتابعيا وذلك بتنفيذ الخطوات التالية

```
MOV  AH , 14H        ; Read a record
LEA  DX , FCBname    ; sequentially
INT  21H             ; Call DOS
```

تقوم تعليمة القراءة باستخدام المعلومات في FCB لتحديد بداية السجل في DTA وكنتيجة لتنفيذ هذه التعليمة يتم ارجاع القيمة (00) الى المسجل AL للدلالة على نجاح عملية القراءة ، (01) نهاية الملف (لا يوجد سجلات) ، (02) لا يوجد مكان كافي في DTA لقراءة السجل ، (03) نهاية الملف .

والبرنامج التالي يوضح كيفية قراءة سجلات الملف المستحدث سابقا وعرض محتويات هذه السجلات على الشاشة.

```
OPENM    MACRO
                    LOCAL LL20
                    LOCAL LL30
                    LEA DX,FCBREC
;REQUEST TO OPEN
                    MOV AH,0FH
                    INT 21H
                    CMP AL,00
;FILE FOUND?
                    JNZ LL20
                    MOV FCBRCSZ,RECLEN
;SET RECORD LENGTH
                    MOV AH,1AH
                    LEA DX,NAMEFLD
;SET ADDRESS OF DTA
                    INT 21H
                    JMP LL30
LL20:
                    MOV ENDCDE,01
;ERROR MESSAGE
                    LEA DX,OPENMSG
                    ERRM
LL30:
                    ENDM
READM    MACRO
                    LOCAL LL20
                    LOCAL LL30
                    MOV AH,14H
;REQUEST TO READ
                    LEA DX,FCBREC
                    INT 21H
                    CMP NAMEFLD,1AH
;END OF FILE MARKER
                    JNE LL20
                    MOV ENDCDE,01
                    JMP LL30
LL20:
                    CMP AL,00
;NORMAL READ
                    JZ LL30
                    MOV ENDCDE,01
                    CMP AL,01
;END OF FILE
                    JZ LL30
                    LEA DX,READMSG
                    ERRM
LL30:
                    ENDM
DISPM    MACRO
                    LOCAL LL20
                    LOCAL LL30
                    MOV AH,09
;REQUEST TO DISPLAY
                    LEA DX,NAMEFLD
```

```
                                INT 21H
                                CMP ROW,20
                ;BOTTOM OF SCREEN
                                JAE LL20
                                INC ROW
                                JMP LL30
                LL20:
                                MOV AX,06001H
                                SCRM
                                CURSM
                LL30:
                                ENDM
                SCRM            MACRO
                                MOV BH,1EH
                ;SET COLOR
                                MOV CX,0000
                                MOV DX,184FH
                ;REQUEST TO SCROLL
                                INT 10H
                                ENDM
                CURSM   MACRO
                                MOV AH,02
                                MOV BH,00
                                MOV DH,ROW
                                MOV DL,00
                                INT 10H
                                ENDM
                ERRM            MACRO
                                MOV AH,09
                                INT 21H
                                ENDM
                ;---------------------------------------------
                -
0000            SSEG            SEGMENT PARA STACK 'STACK'
0000  0050[                     DW 80 DUP(?)
        ????
                  ]

00A0            SSEG            ENDS
                ;---------------------------------------------
                --
0000            DSEG            SEGMENT PARA 'DATA'
0000            FCBREC  LABEL BYTE          ;FCB FOR DISK FILE
0000  .03       FCBDRIV DB 3                ;DRIVE C
0001  4E 41 4D 45 46 49  FCBNAME DB 'NAMEFILE'  ;FILE NAME
      4C 45
0009  44 41 54  FCBEXT  DB 'DAT'            ;EXTENSION
000C  0000      FCBBLK  DW 0000             ;CURRENT BLOCK NO
000E  0000      FCBRCSZ DW 0000             ;LOGICAL REC. SIZE
0010  00000000                  DD ?                ;DOS FILE
0014  0000                      DW ?                ;DOS DATE
0016  000000000000000000        DT ?                ;DOS RESERVE
                D
      00
```

```
0020   00                              FCBSQRC DB 00                    ;CURRENT REC NO
0021   00000000                                    DD ?                        ;RELATIVE RE
                                       C NO
= 0020                                 RECLEN  EQU 32                   ;RECORD LENGTH
0025   0020[                           NAMEFLD DB RECLEN DUP(' '),13, 10, '$'
            20
                      ]
       0D 0A 24
0048   00                              ENDCDE  DB 00
0049   2A 2A 2A 20 4F 50               OPENMSG DB '*** OPEN ERROR ***'
       45 4E 20 45 52 52
       4F 52 20 2A 2A 2A
005B   2A 2A 2A 20 52 45               READMSG DB '*** READ ERROR ***'
       41 44 20 45 52 52
       4F 52 20 2A 2A 2A
006D   00                              ROW              DB 00
006E                                   DSEG             ENDS
                                       ;----------------------------------------------
                                       -------
0000                                   CSEG             SEGMENT PARA 'CODE'
0000                                   BEGIN   PROC FAR
                                                        ASSUME CS:CSEG,DS:DSEG,SS:SSEG,
                                       ES:DSEG
0000   1E                                               PUSH DS
0001   2B C0                                            SUB AX,AX
0003   50                                               PUSH AX
0004   B8 ---- R                                        MOV AX,DSEG
0007   8E D8                                            MOV DS,AX
0009   8E C0                                            MOV ES,AX
000B   B8 0600                                          MOV AX,0600H
                                                        SCRM
000E   B7 1E                      1                     MOV BH,1EH
0010   B9 0000                    1                     MOV CX,0000
0013   BA 184F                    1                     MOV DX,184FH
0016   CD 10                      1                     INT 10H
                                                        CURSM
0018   B4 02                      1                     MOV AH,02
001A   B7 00                      1                     MOV BH,00
001C   8A 36 006D R               1                     MOV DH,ROW
0020   B2 00                      1                     MOV DL,00
0022   CD 10                      1                     INT 10H
                                                        OPENM
0024   8D 16 0000 R               1                     LEA DX,FCBREC
0028   B4 0F                      1                     MOV AH,0FH
002A   CD 21                      1                     INT 21H
002C   3C 00                      1                     CMP AL,00
002E   75 11                      1                     JNZ ??0000
0030   C7 06 000E R 0020          1                     MOV FCBRCSZ,RECLEN
0036   B4 1A                      1                     MOV AH,1AH
0038   8D 16 0025 R               1                     LEA DX,NAMEFLD
003C   CD 21                      1                     INT 21H
003E   EB 0E 90                   1                     JMP ??0001
0041                              1    ??0000:
0041   C6 06 0048 R 01            1                     MOV ENDCDE,01
```

```
0046   8D 16 0049 R        1                  LEA DX,OPENMSG
004A   B4 09               2                  MOV AH,09
004C   CD 21               2                  INT 21H
004E                       1  ??0001:
004E   80 3E 0048 R 00                        CMP ENDCDE,00
0053   75 64                                  JNZ M10
0055                          LOOP1:
                                              READM
0055   B4 14               1                  MOV AH,14H
0057   8D 16 0000 R        1                  LEA DX,FCBREC
005B   CD 21               1                  INT 21H
005D   80 3E 0025 R 1A     1                  CMP NAMEFLD,1AH
0062   75 08               1                  JNE ??0002
0064   C6 06 0048 R 01     1                  MOV ENDCDE,01
0069   EB 16 90            1                  JMP ??0003
006C                       1  ??0002:
006C   3C 00               1                  CMP AL,00
006E   74 11               1                  JZ ??0003
0070   C6 06 0048 R 01     1                  MOV ENDCDE,01
0075   3C 01               1                  CMP AL,01
0077   74 08               1                  JZ ??0003
0079   8D 16 005B R        1                  LEA DX,READMSG
007D   B4 09               2                  MOV AH,09
007F   CD 21               2                  INT 21H
0081                       1  ??0003:
0081   80 3E 0048 R 00                        CMP ENDCDE,00
0086   75 31                                  JNZ M10
                                              DISPM
0088   B4 09               1                  MOV AH,09
008A   8D 16 0025 R        1                  LEA DX,NAMEFLD
008E   CD 21               1                  INT 21H
0090   80 3E 006D R 14     1                  CMP ROW,20
0095   73 07               1                  JAE ??0004
0097   FE 06 006D R        1                  INC ROW
009B   EB 1A 90            1                  JMP ??0005
009E                       1  ??0004:
009E   B8 6001             1                  MOV AX,06001H
00A1   B7 1E               2                  MOV BH,1EH
00A3   B9 0000             2                  MOV CX,0000
00A6   BA 184F             2                  MOV DX,184FH
00A9   CD 10               2                  INT 10H
00AB   B4 02               2                  MOV AH,02
00AD   B7 00               2                  MOV BH,00
00AF   8A 36 006D R        2                  MOV DH,ROW
00B3   B2 00               2                  MOV DL,00
00B5   CD 10               2                  INT 10H
00B7                       1  ??0005:
00B7   EB 9C                                  JMP LOOP1
00B9                          M10:
00B9   CB                                     RET
00BA                          BEGIN   ENDP
00BA                          CSEG            ENDS
                                              END BEGIN
```

المعالجة العشوائية

RANDOM PROSESSING

استعرضنا فيما سبق عملية القراءة التتابعية وعند استخدام هذه الطريقة يتم استعراض (قراءة) عدد من السجلات وذلك من اجل الوصول الى سجل معين فلو اردنا مثلا الوصول الى السجل رقم ١٠٠ فلا بد من قراءة كافة السجلات التي تسبق هذا السجل مما يؤثر بدوره على سرعة المعالجة وبشكل سلبي اما المعالجة العشوائية فتتلخص في تحديد مفتاح السجل المطلوب (رقم السجل النسبي) (relative number) حيث يتم تحويل هذا الرقم الى رقم كتلة (البايت ١٢ – ١٣ في FCB) وترتيب السجل في هذه الكتلة (البايت ٣٢ في FCB) وبهذه الطريقة يتم تحديد موقع السجل الفعلي حيث تتم عملية الانتقال المباشرة الى السجل المطلوب .

وتتضمن عملية المعالجة العشوائية عمليات القراءة والكتابة العشوائية وسوف نوضح هنا كيف تتم هذه العمليات .

القراءة العشوائية Random reading

لقراءة الملف عشوائيا لا بد من تنفيذ الخطوات التالية

١ – فتح الملف (OPEN) وتتم هذه العملية كما في حالة القراءة التتابعية .

٢ – القراءة العشوائية وذلك بتنفيذ الخطوات التالية

```
MOV  AH   , 21H         ; Random reading
LEA  DX   , FCBname  ; request
INT  21H  ; Call DOS
```

حيث يتم تخزين رقم السجل المطلوب في FCB ويتم تحويل هذا الرقم الى رقم كتلة وسجل حيث يتم الوصول الى هذا السجل على القرص المغناطيسي ونقله الى DTA مع التأثير على المسجل AL وذلك بارجاع القيم التالية: (00) عملية القراءة ناجحة ، (01) لا يتوفر السجل المطلوب ، (02) لا يوجد مكان في DTA ، (03) نهاية الملف .

الكتابة العشوائية Random writing

تتم عملية التحضير لهذه العملية كما في عملية القراءة وتستخدم التعليمات التالية

لاجراء عملية الكتابة العشوائية :

```
MOV  AH , 22H          ; Random writing
LEA   DX , FCBname   ; request
INT   21H ; Call DOS
```

فاذا كان رقم السجل المطلوب موجودا يتم تعديله واعادة كتابته في نفس الموقع اما اذا لم يكن موجودا فيتم اضافتها الى الملف وتتم ارجاع القيم التالية الى المسجل AL (00) : عملية الكتابة ناجحة ، (01) القرص ممتلىء ، (02) لا يتوفر مكان في DTA

وفيما يلي سوف نستعرض مثالا يوضح عملية القراءة العشوائية وذلك باستخدام الملف المستحدث سابقا .

```
OPENM     MACRO
                    LOCAL  LL20
                    LOCAL  LL30
                    MOV  AH,0FH
                    LEA  DX,FCBREC
                    INT  21H
                    CMP  AL,00
                    JNZ  LL20
                    MOV  FCBRCSZ,RECLEN
                    MOV  AH,1AH
                    LEA  DX,NAMEFLD
                    INT  21H
                    JMP  LL30
LL20:
                    LEA  DX,OPENMSG
                    ERRM
LL30:
                    ENDM
RECNM     MACRO
                    LOCAL  LL20
                    LOCAL  LL30
                    LOCAL  LL40
                    MOV  AH,09H
                    LEA  DX,PROMPT
                    INT  21H
                    MOV  AH,0AH
; REQUEST TO INPUT
                    LEA  DX,RECDPAR
                    INT  21H
                    CMP  ACTLEN,01
                    JB  LL40
                    JA  LL20
                    SUB  AH,AH
                    MOV  AL,RECDNO
                    JMP  LL30
```

```
LL20:
                    MOV AH,RECDNO
                    MOV AL,RECDNO+1
LL30:
                    AND AX,0F0FH
                    AAD
;CONVERT TO BINARY
                    MOV WORD PTR FCBRNRC,AX
LL40:
                    MOV COL,20
                    CURSM
                    ENDM
READM     MACRO
                    LOCAL LL20
                    MOV ENDCDE,00
                    MOV AH,21H
                    LEA DX,FCBREC
                    INT 21H
                    CMP AL,00
                    JZ LL20
                    LEA DX,READMSG
                    ERRM
LL20:
                    ENDM
DISPM     MACRO
                    MOV AH,09
                    LEA DX,NAMEFLD
                    INT 21H
                    INC ROW
                    MOV COL,00
                    ENDM
CLRM                MACRO
                    MOV AX,0600H
                    MOV BH,41H
;COLOR 07
                    MOV CX,0000
                    MOV DX,184FH
                    INT 10H
                    ENDM
CURSM     MACRO
                    MOV AH,02
                    MOV BH,00
                    MOV DH,ROW
                    MOV DL,COL
                    INT 10H
                    ENDM
ERRM                MACRO
                    MOV AH,09
                    INT 21H
                    INC ROW
                    MOV ENDCDE,01
                    ENDM
```

```
                                    ;---------------------------------------
                                    -------
0000                                CSEG            SEGMENT PARA 'CODE'
                                                    ASSUME CS:CSEG,DS:CSEG,SS:CSEG,
                                    ES:CSEG
0100                                                ORG 100H
0100   EB 7E 90                     BEGIN:  JMP MAIN
                                    ;---------------------------------------
                                    --------
0103                                FCBREC  LABEL BYTE
0103   03                           FCBDRIV DB 03
0104   4E 41 4D 45 46 49            FCBNAME DB 'NAMEFILE'
       4C 45
010C   44 41 54                     FCBEXT  DB 'DAT'
010F   0000                         FCBBLK  DW 0000
0111   0000                         FCBRCSZ DW 0000
0113   00000000                                     DD ?
0117   0000                                         DW ?
0119   000000000000000000                           DT ?
       00
0123   00                                           DB 00


0124   00000000                     FCBRNRC DD 00000000
= 0020                              RECLEN  EQU 32
0128                                RECDPAR LABEL BYTE
0128   03                           MAXLEN  DB 3
0129   00                           ACTLEN  DB ?
012A   0003[                        RECDNO  DB 3 DUP(' ')
           20
                          ]

012D   0020[                        NAMEFLD DB RECLEN DUP(' '),13, 10, '$'
           20
                          ]
       0D 0A 24
0150   2A 2A 2A 20 4F 50            OPENMSG DB '*** OPEN ERROR ***'
       45 4E 20 45 52 52
       4F 52 20 2A 2A 2A
0162   2A 2A 2A 20 52 45            READMSG DB '*** READ ERROR ***'
       41 44 20 45 52 52
       4F 52 20 2A 2A 2A
0174   00                           COL             DB 00
0175   52 45 43 4F 52 44            PROMPT  DB 'RECORD? $'
       3F 20 24
017E   00                           ROW             DB 00
017F   00                           ENDCDE  DB 00
                                    ;---------------------------------------
0180                                MAIN            PROC NEAR
                                                    CLRM
0180   B8 0600                  1                   MOV AX,0600H
0183   B7 41                    1                   MOV BH,41H
0185   B9 0000                  1                   MOV CX,0000
0188   BA 184F                  1                   MOV DX,184FH
018B   CD 10                    1                   INT 10H
                                                    CURSM
018D   B4 02                    1                   MOV AH,02
018F   B7 00                    1                   MOV BH,00
0191   8A 36 017E R             1                   MOV DH,ROW
0195   8A 16 0174 R             1                   MOV DL,COL
0199   CD 10                    1                   INT 10H
```

```
                                                        OPENM
019B  B4 0F                  1                          MOV AH,0FH
019D  8D 16 0103 R           1                          LEA DX,FCBREC
01A1  CD 21                  1                          INT 21H
01A3  3C 00                  1                          CMP AL,00
01A5  75 11                  1                          JNZ ??0000
01A7  C7 06 0111 R 0020      1                          MOV FCBRCSZ,RECLEN
01AD  B4 1A                  1                          MOV AH,1AH
01AF  8D 16 012D R           1                          LEA DX,NAMEFLD
01B3  CD 21                  1                          INT 21H
01B5  EB 12 90               1                          JMP ??0001
01B8                         1   ??0000:
01B8  8D 16 0150 R           1                          LEA DX,OPENMSG
01BC  B4 09                  2                          MOV AH,09
01BE  CD 21                  2                          INT 21H
01C0  FE 06 017E R           2                          INC ROW



01C4  C6 06 017F R 01        2                          MOV ENDCDE,01
01C9                         1   ??0001:
01C9  80 3E 017F R 00                                   CMP ENDCDE,00
01CE  74 01                                             JZ LOOP1
01D0  C3                                                RET
01D1                             LOOP1:
                                                        RECNM
01D1  B4 09                  1                          MOV AH,09H
01D3  8D 16 0175 R           1                          LEA DX,PROMPT
01D7  CD 21                  1                          INT 21H
01D9  B4 0A                  1                          MOV AH,0AH
01DB  8D 16 0128 R           1                          LEA DX,RECDPAR
01DF  CD 21                  1                          INT 21H
01E1  80 3E 0129 R 01        1                          CMP ACTLEN,01
01E6  72 19                  1                          JB ??0004
01E8  77 08                  1                          JA ??0002
01EA  2A E4                  1                          SUB AH,AH
01EC  A0 012A R              1                          MOV AL,RECDNO
01EF  EB 08 90               1                          JMP ??0003
01F2                         1   ??0002:
01F2  8A 26 012A R           1                          MOV AH,RECDNO
01F6  A0 012B R              1                          MOV AL,RECDNO+1
01F9                         1   ??0003:
01F9  25 0F0F                1                          AND AX,0F0FH
01FC  D5 0A                  1                          AAD
01FE  A3 0124 R              1                          MOV WORD PTR FCBRNRC,AX
0201                         1   ??0004:
0201  C6 06 0174 R 14        1                          MOV COL,20
0206  B4 02                  2                          MOV AH,02
0208  B7 00                  2                          MOV BH,00
020A  8A 36 017E R           2                          MOV DH,ROW
020E  8A 16 0174 R           2                          MOV DL,COL
0212  CD 10                  2                          INT 10H
0214  80 3E 0129 R 00                                   CMP ACTLEN,00
0219  74 3D                                             JE L40
                                                        READM
021B  C6 06 017F R 00        1                          MOV ENDCDE,00
0220  B4 21                  1                          MOV AH,21H
0222  8D 16 0103 R           1                          LEA DX,FCBREC
0226  CD 21                  1                          INT 21H
0228  3C 00                  1                          CMP AL,00
022A  74 11                  1                          JZ ??0005
022C  8D 16 0162 R           1                          LEA DX,READMSG
0230  B4 09                  2                          MOV AH,09
```

```
0232   CD 21                 2                  INT 21H
0234   FE 06 017E R          2                  INC ROW
0238   C6 06 017F R 01       2                  MOV ENDCDE,01
023D                         1   ??0005:
023D   80 3E 017F R 00                          CMP ENDCDE,00
0242   75 11                                    JNZ L30
                                                DISPM
0244   B4 09                 1                  MOV AH,09
0246   8D 16 012D R          1                  LEA DX,NAMEFLD
024A   CD 21                 1                  INT 21H
```

```
024C   FE 06 017E R          1                  INC ROW
0250   C6 06 0174 R 00       1                  MOV COL,00
0255                              L30:
0255   E9 01D1 R                                JMP LOOP1
0258                              L40:
0258   C3                                       RET
0259                              MAIN          ENDP
0259                              CSEG          ENDS
                                                END      BEGIN
```

المعالجة العشوائية للكتل

RANDOM BLOCK PROCESSING

تتكون الكتلة من مجموعة من السجلات فاذا توفر الحيز المناسب فإنه يمكن كتابة الكتلة من DTA الى القرص المغناطيسي او قراءة الكتلة وذلك بنقلها من القرص المغناطيسي الى DTA ولاجراء هذا لا بد من تحديد رقم السجل وعدد السجلات في الكتلة الواحدة .

ففي عملية كتابة الكتلة العشوائية يتم تحديد عدد السجلات في الكتلة وذلك بتخزين هذا العدد في المسجل CX وتحديد رقم السجل الاول ثم تنفيذ الخطوات التالية

```
MOV   AH , 28H   ; Request random block write
MOV   CX , Records    ;set number of records
LEA   DX , FCBname
INT   21H   ; Call DOS
```

ونتيجة لتنفيذ هذا يتم ارجاع القيم التالية الى المسجل AL :

(00) عملية كتابة كافة السجلات ناجحة ، (01) لم تتم عملية الكتابة نظرا لعدم توفر حيز كاف على القرص المغناطيسي .

اما قراءة الكتلة فتتم حسب الخطوات التالية

```
MOV   AH , 27H   ;Request random block reading
MOV   CX , records   ; set number of records
LEA   DX , FCBname
INT   21H   ;Call DOS
```

وكنتيجة لتنفيذ امر القراءة يتم ارجاع القيم التالية الى المسجل AL :

(00) عملية القراءة ناجحة ، (01) تمت قراءة آخر سجل في الملف ، (02) قراءة عدد كبير غير محدد من السجلات في DTA ، (03) نهاية الملف .

وفيما يلي برنامجا يوضح المفاهيم الاساسية لعملية قراءة الكتلة العشوائية

```
OPENM     MACRO
                    LOCAL  LL30
                    LOCAL  LL20
                    MOV  AH,0FH
                    LEA  DX,FCBREC
                    INT  21H
                    CMP  AL,00
                    JNZ  LL20
                    MOV  FCBRCSZ,0020H
                    MOV  AH,1AH
                    LEA  DX,DSKRECS
                    INT  21H
                    JMP  LL30
LL20:
                    LEA  DX,OPENMSG
                    ERRM
LL30:
                    ENDM
READM     MACRO
                    MOV  AH,27H
                    MOV  CX,NORECS
                    LEA  DX,FCBREC
                    INT  21H
                    MOV  ENDCODE,AL
                    ENDM
DISPM     MACRO
                    MOV  AH,09
                    LEA  DX,DSKRECS
                    INT  21H
                    ENDM
CLRM                MACRO
                    MOV  AX,0600H
                    MOV  BH,41H
                    MOV  CX,0000
                    MOV  DX,184FH
                    INT  10H
                    ENDM
```

```
CURSM    MACRO
                   MOV AH,02
                   MOV BH,00
                   MOV DH,ROWCTR
                   MOV DL,00
                   INT 10H
                   INC ROWCTR
                   ENDM
ERRM               MACRO
                   MOV AH,09
                   INT 21H
                   MOV ENDCODE,01
                   ENDM
;----------------------------------------
CSEG               SEGMENT PARA 'CODE'
                   ASSUME CS:CSEG,DS:CSEG,SS:CSEG,
ES:CSEG


0100                                           ORG 100H
0100  E9 054D R                 BEGIN:   JMP MAIN
                                ;--------------------------------
0103                            FCBREC   LABEL BYTE
0103  03                        FCBDRIV DB 03
0104  4E 41 4D 45 46 49         FCBNAME DB 'NAMEFILE'
      4C 45
010C  44 41 54                  FCBEXT   DB 'DAT'
010F  0000                      FCBBLK   DW 0000
0111  0000                      FCBRCSZ DW 0000
0113  00000000                  FCBFLZ   DD ?
0117  0000                                     DW ?
0119  000000000000000000                       DT ?
      00
0123  00                                       DB 00
0124  00000000                  FCBRNRC DD 00000000
0128  0400[                     DSKRECS DB 1024 DUP(?),'$'
          ??
                          ]
      24
0529  00                        ENDCODE DB 00
052A  0019                      NORECS   DW 25
052C  2A 2A 2A 4F 50 45         OPENMSG DB '***OPEN ERROR***'
      4E 20 45 52 52 4F
      52 2A 2A 2A
053C  2A 2A 2A 52 45 41         READMSG DB '***READ ERROR***'
      44 20 45 52 52 4F
      52 2A 2A 2A
054C  00                        ROWCTR   DB 00
                                ;--------------------------------
                                ----
054D                            MAIN              PROC NEAR
                                                  .SALL
                                                  CLRM
                                                  CURSM
```

```
0594   80 3E 0529 R 00            OPENM
0599   75 17                      CMP ENDCODE,00
                                  JNZ L30
                                  READM
                                  DISPM
05B2                   L30:
05B2   C3                         RET
05B3                   MAIN       ENDP
05B3                   CSEG       ENDS
                                  END BEGIN
```

العمليات المتعلقة بالقرص المغناطيسي

هناك بعض العمليات الخاصة والمتعلقة بوحدة الاقراص المغناطيسية نورد منها ما يلي

– اقفال الوحدة Disk Reset

حيث تؤدي هذه العملية الى الغاء كافة المعلومات المخزنة في الذاكرة المؤقتة (RAM) ويمكن تنفيذ هذه العملية حسب الخطوات التالية

```
MOV   AH , 0DH  ; Request to reset a disk
INT   21H   ; Call DOS
```

– تغير وحدة الاقراص المغنطة Set default disk drive

قد يتطلب الامر تغير العمل على وحدة الاقراص ونقله الى وحدة اخرى حيث ترقم الوحدات ابتداء من الرقم صفر والذي يدل على الوحدة A , 1 الوحدة B وهكذا ويتم تحديد وحدة الاقراص المطلوبة وذلك حسب الخطوات التالية

```
MOV   AH , OEH   ; Request set drive
MOV  DL , 01   ; drive   B
INT   21H   ; Call DOS
```

– البحث في الفهرس Search for directory entries

قد يتطلب الامر البحث عن مدخلة معينة في الفهرس ولعمل هذا يمكن تنفيذ الخطوات التالية

```
MOV   AH , 11H   ; Request first entry
LEA   DX , FCBname ;
INT  21H   ; Call DOS
```

وعند ايجاد المدخلة المطلوبة يتم ارجاع القيمة (00) الى المسجل AL وإلا يتم ارجاع القيمة (FF) واذا كان هناك اكثر من مدخلة كالبحث مثلا عن كل الملفات من نوع ASM (ASM . *) يمكن تنفيذ الخطوات التالية

```
MOV   AH , 12H   ; Request next entry
LEA   DX , FCBname
INT   21H
```

– حذف الملف Delete a file

يتم حذف الملف وذلك من خلال تنفيذ الخطوات التالية

```
MOV   AH , 13H   ; Request to delete a file
LEA   DX , FCBname
INT   21H
```

فاذا وُجد الملف يحذف ويتم ارجاع القيمة (00) الى المسجل AL وإلا يتم ارجاع القيمة FF

– تغيير اسم الملف Rename file

يتم تغيير اسم الملف وذلك حسب الخطوات التالية

```
MOV   AH , 17H   ;Request to rename
LEA   DX , FCBname
INT   21H
```

– معرفة وحدة الاقراص العاملة Get default drive

وتتم هذه العملية باستخدام الخطوات التالية

```
MOV AH , 19H ;   Get default drive
INT 21H
```

وفيما يلي برنامجا يوضح كيفية حذف ملف ولتنفيذ هذا البرنامج استخدم ملفات لست بحاجة اليها (نسخ من ملفات اخرى) .

```
DISPM    MACRO
;DISPLAY LINE
                 MOV AH,09
                 INT 21H
                 ENDM
CHARM    MACRO
;DISPLAY CHARACTER
                 MOV AH,02
                 INT 21H
                 ENDM
```

```
DISKM   MACRO
                    LOCAL LL10
;READ DIRECTORY ENTRY
                    MOV DX,5CH
                    INT 21H
                    CMP AL,0FFH
                    JNE LL10
                    PUSH AX
                    LEA DX,ENDMSG
                    DISPM
                    POP AX
LL10:
                    ENMD
;PROGRAM TO DELETE SELECTED FILE
;-------------------------------
CSEG                SEGMENT PARA 'CODE'
                    ASSUME CS:CSEG,DS:CSEG,SS:CSEG,ES:CSEG
                    ORG 100H
BEGIN:  JMP MAIN
;ASSUME DEFAULT DRIVE
;ENTER AS *.*, *.LST ETC.
;-----------------------------------------------
TAB                 EQU 09
LF                  EQU 10
CR                  EQU 13
CRLF                DB CR, LF, '$'
DELMSG  DB TAB, 'ERASE', '$'
ENDMSG  DB CR, LF, 'NO MORE DIRECTORY ENTRIES',CR,LF,'$'
ERRMSG  DB 'WRITE PROTECTED DISK','$'
PROMPT  DB 'Y :ERASE, N:KEEP, RET:EXIT',CR,LF,'$'
;---------------------------------------------------
MAIN                PROC NEAR
                    MOV AH,11H    ;LOCATE FIRST ENTRY
                    DISKM
                    CMP AL,0FFH
                    JE LL90       ;IF NO ENTRIES EXIT
                    LEA DX,PROMPT
                    DISKM
LL20:
                    LEA DX,DELMSG
                    DISKM
                    MOV CX,11         ;11 CHARACTERS
                    MOV SI,81H        ;START OF FILE NAME
LL30:
                    MOV DL,[SI]       ;GET CHAR FOR DISPLAY
                    CHARM
                    INC SI            ;NEXT CHAR
                    LOOP LL30
                    MOV DL,'?'
                    CHARM
                    MOV AH,01         ;ACCEPT FIRST CHAR.
                    INT 21H
                    CMP AL,0DH        ;RETURN CHAR ?
```

```
          JE LL90            ;YES EXIT
          OR AL,00100000B    ;LOWERCASE
          CMP AL,'Y'         ;DELETE REQUESTED?
          JNE LL50           ;NO BYPASS
          MOV AH,13H         ;YES DELETE
          MOV DX,80H
          INT 21H
          CMP AL,0           ;WAS DELETED VALID?
          JZ LL50
          LEA DX,ERRMSG      ;NO ERROR
          DISPM
          JMP LL90
LL50:
          LEA DX,CRLF        ;RETURN/LINE FEED
          DISPM
          MOV AH,12H
          DISKM
          CMP AL,0FFH        ;ANY MORE
          JNE LL20
LL90:
          RET
MAIN      ENDP
CSEG      ENDS
          END BEGIN
```

الاقراص الممغنطة وبرمجيات الادخال والاخراج
BIOS DISK I/O

تستخدم برمجيات الادخال والاخراج (Basic Input Output System) لاجراء كافة العمليات على وحدات الاقراص الممغنطة وذلك باستخدام المسجلات الخاصة كما يلي

المسجل	الهدف
AH	العملية المراد اجراؤها (Read , write , verify , format)
AL	عدد المقاطع (Sectors)
CH	رقم المسار
CL	رقم القاطع الاول
DH	الوجه المستخدم (صفر أو ١ للاقراص المرنة)
DL	رقم الوحدة 0 - A , 1 - B
ES: BX	عنوان منطقة التخزين المخصصة للادخال والاخراج

ولاجراء العمليات اللازمة على القرص باستخدام BIOS يستخدم الاعتراض 13H وذلك بتخزين التعليمة المراد تنفيذها في المسجل AH ثم استدعاء برمجيات الادخال والاخراج باستخدام الاعتراض 13H وفيما يلي اهم التعليمات المستخدمة والقيمة المراد تخزينها في AH لتنفيذها .

القيمة	العملية
00	Reset diskette
01	Read diskette Status
02	Read Sectors
03	Write Sectors
04	Verify Sector
05	Format track

واذا تمت هذه العمليات بنجاح فان قيمة CF , AH ستصبح صفر اما اذا لم تتم احدى هذه العمليات بنجاح فان قيمة CF ستصبح مساوية للصفر وسوف يتم ارجاع قيمة معينة الى AH تدل على نوع الخطأ الحادث نتيجة التنفيذ .

فمثلا لقراءة محتويات قاطع في المنطقة areal يمكن تنفيذ العمليات التالية

```
MOV   AH  , 02    ; request to read
MOV   AL  , 01    ;one sector
LEA   BX  , areal
MOV   CH  , 05    ;track 5
MOV   CL  , 03    ; sector 3
MOV   DH , 00     ;  head 0
MOV   DL  , 01    ; drive B
INT   13H         ;Call BIOS
```

والبرنامج التالي يوضح عملية قراءة قاطع باستخدام BIOS

```
ADDRM   MACRO
                    LOCAL LL20
                    LOCAL LL90
;MACRO TO CALCULATE NEXT DISK ADDRESS
;-----------------------------------
                    MOV CX,CURADR
;GET TRACK/SECTOR
                    CMP CL,10
                    JNE LL90
                    CMP SIDE,00
;BYPASS IF SIDE 0
                    JE LL20
                    INC CH
LL20:
                    XOR SIDE,01
;CHANGE SIDE
                    MOV CL,01
                    MOV CURADR,CX
LL90:
                    ENDM
DISPM   MACRO
                    MOV AH,40H
                    MOV BX,01
                    MOV CX,512
                    LEA DX,RECDIN
                    INT 21H
                    ENDM
SCRM                MACRO
                    MOV AX,0600H
                    MOV BH,1EH
                    MOV CX,0000
                    MOV DX,184FH
                    INT 10H
                    ENDM
CURSM   MACRO
                    MOV AH,02
                    MOV BH,00
                    MOV DX,0000
                    INT 10H
                    ENDM
ERRM                MACRO
                    MOV AH,40H
                    MOV BX,01
                    MOV CX,18
                    LEA DX,READMSG
                    INT 21H
                    ENDM
;--------------------------------------
0000                    CSEG        SEGMENT PARA 'CODE'
                                    ASSUME CS:CSEG,DS:CSEG,SS:CSEG,
                        ES:CSEG
0100                                ORG 100H
0100   E9 031C R        BEGIN:  JMP MAIN
                        ;--------------------------------------
```

```
0103  0200[
          20
                          ]

0303  00                          ENDCDE   DB 00
0304  0304                        CURADR   DW 0304H ;BEGIN. TR/SECT.
0306  0501                        ENDADR   DW 0501H ;ENDING TR/SECT.
0308  2A 2A 2A 20 52 45           READMSG  DB '*** READ ERROR ***$'
      41 44 20 45 52 52
      4F 52 20 2A 2A 2A
      24
031B  00                          SIDE             DB 00
                                  ;----------------------------------------
031C                              MAIN             PROC NEAR
031C  B8 0600                                      MOV AX,0600H
031F                              LOOP1:
                                                   SCRM
031F  B8 0600                 1                    MOV AX,0600H
0322  B7 1E                   1                    MOV BH,1EH
0324  B9 0000                 1                    MOV CX,0000
0327  BA 184F                 1                    MOV DX,184FH
032A  CD 10                   1                    INT 10H
                                                   CURSM
032C  B4 02                   1                    MOV AH,02
032E  B7 00                   1                    MOV BH,00
0330  BA 0000                 1                    MOV DX,0000
0333  CD 10                   1                    INT 10H
                                                   ADDRM
0335  8B 0E 0304 R            1                    MOV CX,CURADR
0339  80 F9 0A                1                    CMP CL,10
033C  75 14                   1                    JNE ??0001
033E  80 3E 031B R 00         1                    CMP SIDE,00
0343  74 02                   1                    JE ??0000
0345  FE C5                   1                    INC CH
0347                          1   ??0000:
0347  80 36 031B R 01         1                    XOR SIDE,01
034C  B1 01                   1                    MOV CL,01
034E  89 0E 0304 R            1                    MOV CURADR,CX
0352                          1   ??0001:
0352  8B 0E 0304 R                                 MOV CX,CURADR
0356  8B 16 0306 R                                 MOV DX,ENDADR
035A  3B CA                                        CMP CX,DX
035C  74 1A                                        JE LL90
035E  E8 0379 R                                    CALL READP
0361  80 3E 0303 R 00                              CMP ENDCDE,00
0366  75 10                                        JNZ LL90
                                                   DISPM
0368  B4 40                   1                    MOV AH,40H
036A  BB 0001                 1                    MOV BX,01
036D  B9 0200                 1                    MOV CX,512
0370  8D 16 0103 R            1                    LEA DX,RECDIN
0374  CD 21                   1                    INT 21H
0376  EB A7                                        JMP LOOP1
0378                              LL90:
```

The first line of the listing reads as follows (wide line):

```
0103  0200[                       RECDIN   DB 512 DUP(' ')   ;INPUT AREA
```

```
0378  C3                                    RET
0379                         MAIN    ENDP
                             ;READ DISK SECTOR
                             ;---------------
0379                         READP   PROC NEAR
0379  B0 01                                 MOV AL,01;NO OF SECTORS
037B  B4 02                                 MOV AH,02
037D  8D 1E 0103 R                          LEA BX,RECDIN
0381  8B 0E 0304 R                          MOV CX,CURADR
0385  8A 36 031B R                          MOV DH,SIDE
0389  B2 01                                 MOV DL,01
038B  CD 13                                 INT 13H
038D  80 FC 00                              CMP AH,00
0390  74 13                                 JZ ZZ90
0392  C6 06 0303 R 01                       MOV ENDCDE,01
                                            ERRM
0397  B4 40                 1               MOV AH,40H
0399  BB 0001               1               MOV BX,01
039C  B9 0012               1               MOV CX,18
039F  8D 16 0308 R          1               LEA DX,READMSG
03A3  CD 21                 1               INT 21H
03A5                         ZZ90:
03A5  FF 06 0304 R                          INC CURADR
03A9  C3                                    RET
03AA                         READP   ENDP
03AA                         CSEG           ENDS
                                            END  BEGIN
```

الوحدة الثالثة عشرة

ربط البرامج

- ربط برامج لغة التجميع
- ربط برامج التجميع بلغات البرمجة ذات المستوى العالي

ربط البرامج

PROGRAM LINKING

استعرضنا سابقا كيفية كتابة وتنفيذ برنامج التجميع وكيفية كتابة البرامج الفرعية والمكروية وكيفية استدعائها من خلال البرنامج الرئيسي وقد كانت عمليات ترجمة البرنامج تتم بشكل متكامل بحيث تتم عملية ترجمة البرنامج الفرعي والرئيسي معا لتكوين البرنامج الهدفي.

وسوف نستعرض في هذا الباب كيفية ربط برامج مختلفة بعد ترجمة كل برنامج بشكل منفصل لما لهذه العملية من اهمية كبيرة - يمكن تلخيصها في الفوائد التالية:

- تجزئة البرنامج الى مجموعة من البرامج يسهل مراجعتها لاكتشاف الاخطاء ان وجدت .
- امكانية ربط البرامج والمكتوبة بلغات مختلفة مما يتيح امكانية استخدام الامكانيات التي توفرها لغات البرمجة ذات المستوى العالي والمتدني .
- امكانية توزيع الاجزاء المختلفة للبرنامج على اشخاص متعددين وذلك للعمل كفريق واحد ثم ربط هذه الاجزاء معا.

هذا وسوف نستعرض هنا كيفية ربط برامج لغة التجميع والبرامج المكتوبة بلغات ذات مستوى عال مع برنامج التجميع .

ربط برامج التجميع

استعرضنا سابقا كيفية استدعاء البرنامج الفرعي باستخدام التعليمة CALL وكانت عملية الاستدعاء تتم داخل القاطع الواحد (Intrasegment CALL) وذلك باستخدام الاستدعاء القريب (CALL NEAR) اذا كانت الازاحة في العنوان ضمن (128 byte-) او الاستدعاء البعيد (CALL FAR) اذا كانت قيمة الازاحة اكبر من هذه القيم .

وكنتيجة لتنفيذ امر الاستدعاء هذا فان القيمة المخزنة في IP يتم الاحتفاظ بها في SI وقيمة العنوان المراد نقل التحكم اليه يتم تخزينها في المسجل IP حيث تحدد هذه القيمة من خلال التعليمة نفسها فمثلا تعليمة الاستدعاء هذه تمثل كما يلي

E8 2000

حيث يدل الرمز E8 على شيفرة العملية (CALL) اما الرقم 2000 فهو عنوان

التعليمة المراد نقل التحكم اليها داخل القاطع الواحد .

اما عملية الاستدعاء من خارج القاطع الواحد (Intersegment CALL) فتتم من خلال استخدام التعليمة CALL حيث يتم بواسطة هذه التعليمة استدعاء برنامج مستقل (تمت ترجمته بشكل مستقل) ويختلف هذا النوع من الاستدعاء عن الاستدعاء السابق في انه دائما من النوع البعيد (FAR) ويتم في هذا الاستدعاء الاحتفاظ بمحتوى IP , CX ويأخذ هذا الاستدعاء الشكل التالي

9A 0002 AF04

حيث يدل الرمز 9A على شيفرة التعليمة CALL اما المعاملات 0002 فيتم الاحتفاظ بها في IP اما المعامل الثاني فيخزن في CX وذلك من اجل الوصول الى العنوان الحقيقي

Code segment	04AF0
offset in IP	02000
Effective address	04CF0

(لاحظ ان العناوين تخزن بشكل معكوس Reverse)

وعند استخدام امر الاستدعاء هذا يجب اتباع الخطوات التالية

- تتم ترجمة كل من البرنامج المستدعي والمستدعى على حدة .
- استخدام التعليمة EXTRN في البرنامج المستدعي لاخبار المترجم عن وجود برنامج آخر يجب استدعاؤه من خلال هذا البرنامج حيث تأخذ هذه التعليمة الشكل التالي

EXTRN name : type

حيث يدل name على اسم البرنامج المستدعى و Type تحدد نوع القفز (FAR)

- استخدام التعليمة PUBLIC لاخبار المترجم بأن هذا البرنامج يُستدعى من برنامج آخر . وتأخذ هذه التعليمة الشكل التالي

PUBLIC symbol

حيث يحدد symbol اسم هذا البرنامج (يطابق اسم البرنامج في EXTRN)

- عند اجراء عملية الربط (LINK) يتم دمج البرنامج الهدفي (object) الخاص بالبرنامج المستدعي مع البرنامج الهدفي الخاص بالبرنامج المستدعى وذلك لتكوين برنامج تنفيذي واحد كما يلي

object modules [.OBJ] : CALL OBJ + CALLEDOBJ

والمثال التالي يوضح هذه الامور

```
                                          EXTRN CALLEDP:FAR
0000                              SSEG            SEGMENT PARA STACK 'STACK'
0000    0040[                                     DW 64 DUP(?)
           ????
                        ]

0080                              SSEG            ENDS
0000                              DSEG            SEGMENT PARA 'DATA'
0000    0140                      Q1              DW 0140H
0002    2500                      P1              DW 2500H
0004                              DSEG            ENDS
                                  ;--------------------------------
0000                              CSEG            SEGMENT PARA 'CODE'
0000                              BEGIN   PROC FAR
                                                  ASSUME CS:CSEG,DS:DSEG,SS:SSEG
0000    1E                                        PUSH DS
0001    2B C0                                     SUB AX,AX
0003    50                                        PUSH AX
0004    B8 ---- R                                 MOV AX,DSEG
0007    8E D8                                     MOV DS,AX
0009    A1 0002 R                                 MOV AX,P1
000C    8B 1E 0000 R                              MOV BX,Q1
0010    9A 0000 ---- E                            CALL CALLEDP
0015    CB                                        RET
0016                              BEGIN   ENDP
0016                              CSEG            ENDS
                                                  END BEGIN
```

هذا ويمكن استخدام PUBLIC لغرض آخر في البرنامج ألا وهو اشتراك البرنامج المستدعي والمستدعى في قاطع تعليمات واحد وذلك من خلال استخدام هذه التعليمة في قاطع التعليمات كما يلي

CODESG SEGMENT PARA PUBLIC 'CODE'

وذلك في البرنامج المستدعي والمستدعى والبرنامج التالي يوضح هذا

```
                                          EXTRN CALLEDP:FAR
0000                              SSEG            SEGMENT PARA STACK 'STACK'
0000    0040[                                     DW 64 DUP(?)
           ????
                        ]

0080                              SSEG            ENDS
0000                              DSEG            SEGMENT PARA 'DATA'
0000    0140                      Q1              DW 0140H
0002    2500                      P1              DW 2500H
0004                              DSEG            ENDS
                                  ;--------------------------------
0000                              CSEG            SEGMENT PARA PUBLIC 'CODE'
0000                              BEGIN   PROC FAR
                                                  ASSUME CS:CSEG,DS:DSEG,SS:SSEG
0000    1E                                        PUSH DS
0001    2B C0                                     SUB AX,AX
0003    50                                        PUSH AX
0004    B8 ---- R                                 MOV AX,DSEG
0007    8E D8                                     MOV DS,AX
0009    A1 0002 R                                 MOV AX,P1
000C    8B 1E 0000 R                              MOV BX,Q1
0010    9A 0000 ---- E                            CALL CALLEDP
0015    CB                                        RET
0016                              BEGIN   ENDP
0016                              CSEG            ENDS
                                                  END BEGIN
```

```
0000                                   CSEG              SEGMENT PARA 'CODE'
0000                                   CALLEDP PROC FAR
                                                         ASSUME CS:CSEG
                                                         PUBLIC CALLEDP
0000   F7 E3                                             MUL BX
0002                                   CALLEDP ENDP
0002                                   CSEG              ENDS
                                                         END CALLEDP
```

```
 Start  Stop   Length Name                      Class
 00000H 0007FH 00080H SSEG                      STACK
 00080H 00083H 00004H DSEG                      DATA
 00090H 000A5H 00016H CSEG                      CODE
 000B0H 000B1H 00002H CSEG                      CODE

Program entry point at 0009:0000
```

لاحظ من خلال ملف المسح (Map - file) انه يتوفر فقط قاطع تعليمات واحد (انظر الى البرنامج السابق ولاحظ الفروقات) .

وهناك استخدام آخر لتعليمة PUBLIC حيث يمكن استخدامها للمشاركة في البيانات المعرفة من قبل البرنامج المستدعي وذلك لاستخدامها في البرنامج المستدعى والبرنامج التالي يوضح كيفية استخدام PUBLIC لهذا الغرض حيث يتم الاعلان عن قاطع البيانات ايضا باستخدام PUBLIC .

```
                                             EXTRN CALLEDP:FAR
                                                     PUBLIC Q1,P1
0000                              SSEG               SEGMENT PARA STACK 'STACK'
0000   0040[                                         DW 64 DUP(?)
           ????
                           ]

0080                              SSEG               ENDS
0000                              DSEG               SEGMENT PARA 'DATA'
0000   0140                       Q1                 DW 0140H
0002   2500                       P1                 DW 2500H
0004                              DSEG               ENDS
                                  ;-------------------------------------
0000                              CSEG               SEGMENT PARA PUBLIC 'CODE'
0000                              BEGIN    PROC FAR
                                                     ASSUME CS:CSEG,DS:DSEG,SS:SSEG
0000   1E                                            PUSH DS
0001   2B C0                                         SUB AX,AX
0003   50                                            PUSH AX
0004   B8 ---- R                                     MOV AX,DSEG
0007   8E D8                                         MOV DS,AX
0009   A1 0002 R                                     MOV AX,P1
000C   8B 1E 0000 R                                  MOV BX,Q1
0010   9A 0000 ---- E                                CALL CALLEDP
0015   CB                                            RET
0016                              BEGIN    ENDP
0016                              CSEG               ENDS
                                                     END BEGIN
```

```
                                      EXTRN Q1:WORD,P1:WORD
0000                      CSEG        SEGMENT PARA PUBLIC 'CODE'
0000                      CALLEDP PROC FAR
                                      ASSUME CS:CSEG
                                      PUBLIC CALLEDP
0000  A1 0000 E                       MOV AX,P1
0003  8B 1E 0000 E                    MOV BX,Q1
0007  F7 E3                           MUL BX
0009                      CALLEDP ENDP
0009                      CSEG        ENDS
                                      END CALLEDP
```

لاحظ من خلال البرنامج ان البيانات Q1 , P1 عرفت في البرنامج الرئيسي (المستدعي) باستخدام PUBLIC وقد تم استخدام هذه البيانات في البرنامج المستدعي.

ربط برنامج التجميع ببرامج ذات مستوى عال

تمتاز لغات البرمجة ذات المستوى العالي بسهولتها سواء بالكتابة او بمراجعة البرنامج نظرا لاستخدامها تعليمات قريبة من اللغة الانجليزية ولكن قد يتطلب الامر استخدام برامج التجميع لما تمتلكه هذه البرامج من القوة والفعالية وذلك لتنفيذ بعض العمليات والتي قد يصعب تنفيذها باستخدام البرامج ذات المستوى العالي .

ولتحقيق هذا يمكن كتابة البرنامج الرئيسي باستخدام احدى لغات البرمجة ذات المستوى العالي (باسكال مثلا) ، حيث يقوم هذا البرنامج باستدعاء برنامج فرعي مكتوب بلغة التجميع .

ولعمل هذا يتم كتابة برنامج التجميع بشكل منفصل حيث يترجم للحصول على البرنامج الهدفي (object code) والذي يتم استدعاؤه من خلال برنامج باسكال مثلا.

وفيما يلي برنامج مكتوب بلغة باسكال حيث يقوم هذا البرنامج باستدعاء برنامج فرعي مكتوب بلغة التجميع . لاحظ استخدام EXTERNAL عند الاعلان عن البرنامج الفرعي متبوعة باسم البرنامج الهدفي (Assembly object code)

```
PROGRAM ZZ(INPUT,OUTPUT);
PROCEDURE MCRSR(ROW,COL :INTEGER);EXTERNAL 'A:LINK1.OBJ';
VAR
     TROW,TCOL :INTEGER;
BEGIN
     WRITE('ENTER CURSOR ROW:  ');
     READLN(TROW);
     WRITE('ENTER CURSOR COLUMN:  ');
     READLN(TCOL);
     MCRSR(TROW,TCOL);
     WRITE('NEW CURSOR LOCATION')
END.
```

```
                                      PUBLIC MCRSR
0000                          CSEG              SEGMENT PARA PUBLIC 'CODE'
                                                ASSUME CS:CSEG
0000                          MCRSR   PROC FAR
= 0008                        ROWPAR  EQU 8
= 0006                        COLPAR  EQU 6
0000   55                                       PUSH BP
0001   8B EC                                    MOV BP,SP
0003   8B 76 08                                 MOV SI,[BP+ROWPAR]
0006   8A 34                                    MOV DH,[SI]
0008   8B 76 06                                 MOV SI,[BP+COLPAR]
000B   8A 14                                    MOV DL,[SI]
000D   B4 02                                    MOV AH,02
000F   2A FF                                    SUB BH,BH
0011   CD 10                                    INT 10H
0013   5D                                       POP BP
0014   CA 0004                                  RET 4
0017                          MCRSR             ENDP
0017                          CSEG              ENDS
                                                END
```

الملاحق

```
POP   DX
POP   CX
POP   BX
POP   AX
      ENDM
```

TAB MACRO column

```
    LOCAL DO-TAB, GET-BH
;;
;; Move cursor to specified column (0 to 79).
;; If it is beyond this position, move to specified column
;; On the next line.
    PUSH  AX              ;; Save affected registers
    PUSH  BX
    PUSH  CX
    PUSH  DX
    POS                   ;; Read column number into DL.
    CMP   DL, COLUMN      ;; Cursor beyond this column?
    JBE  DO-TAB
    LF                    ;; If so, advance to next line
    POS                   ;; And read new cursor position.
DO-TAB: MOV  DL, COLUMN   ;; Read user's column number.
    CMP   DL, 79          ;; Is it too large?
    JNA  GET-BH
    MOV   DL, 79          ;;  If so, use column 79.
GET-BH : VIDEO-STATE      ;; Read active page into BH.

        MOV   AH,2        ;; Select cursor-moving option.
        INT   10H         ;; Move the cursor.
        POP   DX          ;; Restore registers.
        POP   CX
        POP   BX
        POP   AX
        ENDM
```

VIDEO-STATE MACRO

```
;;
;; Read current mode into AL, number of screen columns into AH,
;; and active page into BH.
    MOV   AH, 15          ;; Select video state option.
    INT   10H             ;; Read video state.
    ENDM
```

```
SOUND  MACRO  freq,duration

    ;;
;;  Produce a tone of a specified frequency and duration.
;;  Frequency is in Hertz;; duration is in hundredths of a second.
;;
    MOV    DI, freq
    MOV    BX, duration
    SOUND-DI-BX
      ENDM
```

```
SOUND-DI-BX  MACRO

    LOCAL WAIT, SPKR-ON
;;
;; Produce a tone of a specified frequency and duration.
;; Frequency in Hertz is in DI; duration in hundredths of a
;; Second is in BX
;;
    PUSH   AX            ;;  Save affected registers.
    PUSH   BX
    PUSH   CX
    PUSH   DX
    PUSH   DI

    MOV    AL, 0B6H              ;; Write timer mode register
    OUT    43H, AL
    MOV    DX, 14H
    MOV    AX, 4F38H             ;;  Timer divisior =
    DIV    DI                    ;;  1331000/Frequency
    OUT    42H, AL               .;;  Write ;;Timer 2 count low byte.
    MOV    AL, AH
    OUT    42H, AL               ;; Write Timer 2 count low byte
    IN     AL, 61H               ;; Get current port B setting
    MOV    AH, AL                ;; and save it in AH.
    OR     AL, 3                 ;;  Turn speaker on.
    OUT    61H, AL
WAIT: MOV   CX, 2801             ;; Wait 10 milliseconds.
SPKR-ON:   LOOP  SPKR-ON
    DEC    BX                    ;;  Speaker-on count expired?
    JNZ    WAIT                  ;;  If not, keep speaker on.
    MOV    AL, AH                ;; Recover value of port.
    OUT    61H, AL               ;; Restore registers.
    POP    DI
```

```
SET-TIME MACRO hours, minutes, seconds, hsecs

;;
;; Set the time. Blank operands become zero.
;;
    PUSH  AX            ;; Save affected registers.
    PUSH  CX
    PUSH  DX
    MOV   CH, hours     ;; Read user values.
    MOV   CL, minutes
    MOV   DH, seconds
    MOV   DL, hsecs
    MOV   AH, 2DH       ;; Select time set option.
    INT   21H           ;; Set the time.
    POP   DX            ;; Restore registers.
    POP   CX
    POP   AX
    ENDM
```

```
SHOW-AL MACRO
;;
;; Display the character in AL. keep the cursor where it is
;;
    PUSH  AX            ;; Save affected registers.
    PUSH  BX
    PUSH  CX
    PUSH  AX            ;; Preserve input value.
    VIDEO-STATE         ;; Read display page into BH.
    POP   AX            ;; Restore input value.
    MOV   CX, 1         ;; Display just one character.
    MOV   AH, 10        ;; Select display option.
    INT   10H           ;; Display the character.
    POP   CX            ;; Restore registers.
    POP   BX
    POP   AX
    ENDM
```

```
POP    CX              ;; Restore AH.
MOV    AH, CH
POP    DX              ;; Restore other registers.
POP    CX.
ENDM
```

READ-TIME MACRO

```
;;
;; Read the time into CH (hours), CL (minutes), DH (seconds),
;; and DL (1/100) seconds).
    PUSH   AX
    MOV    AH, 2CH     ;; Select time-reading option.
    INT    21H         ;; Read the time.
    POP    AX
           ENDM
```

SET-DS MACRO desg-name

```
;;
;; load the address of the data segment into DS.
;;
    PUSH   AX
    MOV    AX,desg-name
    MOV    DS, AX
    POP    AX
     ENDM
```

SET-ES MACRO eseg-name

```
;;
;; load the address of the extra segment into ES.
;;
    PUSH   AX
    MOV    AX, eseg-name
    MOV    ES, AX
    POP    AX
      ENDM
```

PUSH-REGS MACRO reg-list

```
;;
;; Save registers on the stack.
;;
     IRP     reg, <reg-list>
     PUSH    reg
             ENDM
```

RAND MACRO limit

```
     LOCAL STRIP
;;
;; Generate a random integer between 0 and "limit," inclusive,
;; and return it in AL.
;;
     PUSH  CX          ;; Save affected registers.
     PUSH  DX
     PUSH  AX
     MOV   AH, 0       ;; Read the timer
     INT   1AH
     MOV   AX, DX      ;; Move low count into AX
     MOV   CL, limit   ;; Move limit into CL
;;
;; Strip enough high bits off the dividend (AX) to ensure against
;; divide overflow.
     MOV   DH, 3FH     ;; Set up AND  mask in DH.
     CMP   CL, 64
     JAE   STRIP
     SHR   DH, 1       ;; If limit <64, strip off 3 bits.
     CMP   CL, 32
     JAE   STRIP
     SHR   DH, 1       ;; If limit <32, strip off 4 bits.
     CMP   CL,16
     JAE   STRIP
     SHR   DH,1        ;; If limit <16, strip off 5 bits.
     CMP   CL0
     JAE   STRIP
     SHR   DH,1        ;; If limit <8, strip off 6 bits.
STRIP:     AND AH, DH  ;; Strip off the bits.
     DIV   CL          ;; Divide result in AX by limit in CL.
     MOV   AL, AH      ;; Put remainder in AL.
```

```
PRINT$  TEMP            ;; Restore registers.
POP     CX
POP     DS
        ENDM
```

PRINT$ MACRO string-name

```
;;
;; Display a specifiesd data segment string.
;; First byte in the string holds the character count.
;;
     PUSH   DX
     LEA    DX, string-name
     PRINT$-DX
     POP    DX
            ENDM
```

PRINT$-DX MACRO

```
     LOCAL STRIP, NEXT-C, NO-BLANK
;;
;; Display the data segment string whose offset is in DX.
;; First byte in the string holds the character count.
;;                              ;; Save affected register.
     PUSH  AX
     PUSH  CX
     PUSH  SI                   ;; Clear count register.
     SUB   CX, CX               ;; Put string pointer in SI.
     MOV   SI, DX               ;; Read character count.
     MOV   CL, [SI]             ;; Point to first character.
     INC   SI                   ;; Skip leading blanks.
STRIP: LODSB
     CMP   AL, ' '
     JNE   NO-BLANK
     LOOP  STRIP                ;; Read character
NEXT-C: LODSB                   ;; and display it.
NO-BLANK: PRINT-AL
     LOOP  NEXT-C               ;; Restore registers.
     POP   SI
     POP   CX
     POP   AX
           ENDM
```

POS MACRO

```
;;
;; Read the current row and column numbers into DH and DL,
;; and the cursor mode into CH and CL.
    PUSH  AX            ;; Save affected registers.
    PUSH  BX
    VIDEO-STATE         ;; Read display page into BH.
    MOV   AH, 3         ;; Select cursor-reading option.
    INT   10H           ;; Read the cursor.
    POP   BX            Restore registers.
    POP   AX
    ENDM
```

PRINT-AL MACRO

```
;;
;; Display the character in AL, then advance the cursor.
;;
    PUSH  AX            ;; save affected register
    PUSH  BX
    PUSH  CX
    PUSH  AX            ;; Preserve input value.
    VIDEO -STATE        ;; Read display page into BH.
    POP   AX            ;; Restore input value.
    MOV   CX, 1         ;; Display just one character.
    MOV   AH, 14        ;; Select display option.
    INT   10H           ;; Display the character.
    POP   CX            ;; Restore registers.
    POP   BX
    POP   AX
    ENDM
```

PRINT-NUMBER MACRO

```
    LOCAL TEMP, SAVE
;;
;; Display the signed contents of AX.
;;                      ;; Skip temporary buffer.
    JMP   SAVE
TEMP DB 7 DUP (?)       ;; Save affected registers.
SAVE: PUSH DS
    PUSH  CX            ;; Make DS point to code segment.
    MOV   CX, CS
    MOV   DS, CX        ;; Make the conversion.
    BIN2$ TEMP          ;; Display it.
```

```
    MESSAGE-DX
    ENDM
```

MESSAGE-DX MACRO

```
;;
;; Display a message string in the data segment whose offset
;; is in DX. String must end with a $ character.
    PUSH   AX         ;; Select string display option
    MOV    AH, 9      ;; Display the string
    INT    21H
    POP    AX
           ENDM
```

MOVE-CURSOR MACRO

```
    LOCAL OK, VS
;;
;; Move the cursor to the row and column positions in DH and DL,
;; respectively. If the row exceeds 24 or the column exceeds 79,
;; make it 0.                ;; Save affected registers.
    PUSH   AX
    PUSH   BX                ;; Wrap-around input if needed.
    CMP    DH, 24
    JNA    OK
    SUB    DH, DH
OK: CMP    DL, 79
    JNA    OK
    SUB    DL,DL             ;; Read display page into BH.
VS: VIDEO-STATE              ;; Select cursor-moving option.
    MOV    AH,2              ;; Move the cursor.
    INT    10H               ;; Restore flags and registers.
    POP    BX
    POP    AX
           ENDM
```

POP-REGS MACRO reg-list

```
;;
;;Retrieve registers from the stack
    IRP   reg, <reg-list>
    POP   reg
    ENDM
```

LF MACRO

```
;;
;; Move the cursor to next line (same column position).
;; If it is on the bottom line, move it to the top.
;;
     PUSH    CX
     PUSH    DX
     POS                          ;; Read row number into DH
     INC     DH                   ;; and increment it
     MOVE-CURSOR
     POP     DX
     POP     CX
       ENDM
```

LOCATE MACRO row, col

```
;;
;; Move the cursor to the specified row and column.
;;
     PUSH    DX
     MOV     DH, row
     MOV     DL, col
     MOVE-CURSOR
     POP     DX
       ENDM
```

MAKE-STACK MACRO stack-name

```
;;
;; set up a "standard" stack segment.
;;
stack-name SEGMENT PARA STACK 'STACK'
           DB    64 DUP ('STAK')
stack-name ENDS
           ENDM
```

MESSAGE MACRO string-name

```
;;
;; Display the specified data string segment. String must
;; end with a $ character.
;;
     PUSH    DX
     LEA     DX, DS: string•name
```

```
INKEY  MACRO
;;
;;  Read the next key into AL and display it.
;;                                              ;;  Save registers.
      PUSH    CX
      PUSH    AX                                ;; Select key option.
      MOV     AH, 1                             ;  Read the key.
      INT     21H                               ;; Restore AH.
      POP     CX
      POP     AH, CH                            ;;  Restore CX.
      POP     CX
          ENDM
```

```
INKEY  MACRO
;;
;; Read the next key into AL, but don't display it.
;;                                              ;;  Save registers.
      PUSH     CX
      PUSH     AX                               ;; Select key option.
      MOV      AH,8                             ;  Read the key.
      INT      21H                              ;; Restore AH.
      POP      CX
      POP      AH, CH                           ;;  Restore CX.
      POP      CX
      ENDM
```

```
KEY  MACRO

      LOCAL TEMP, SAVE
;;
;;  Read a sequence of keyboard characters and convert them to
;;  a signed number in AX.
;;                                              ;;  Skip temporary buffer.
       JMP    SAVE
TEMP  DB   7,8 DUP (?)                          ;; Save affected registers.
SAVE:  PUSH    DS
      PUSH      CX                              ;;  Make DSpoint to code segment.
      MOV        CX, CS
      MOV        DS, CX                         ;;  Read string into TEMP.
      IN$       TEMP                            ;;  Make the conversion.
      $2BIN      TEMP + 1                       ;;  Restore registers.
      POP       CX
      POP       DS
                ENDM
```

```
CMP   CX, AX
JA     QUIT
JB     CHECK
CMP   DX, BX
JB     CHECK
QUIT: POP DX
POP        CX
POP        BX
POP        AX
   ENDM
```

HOME MACRO

```
;;
;; Move cursor to upper left-hand corner
;;
    LOCATE 0,0
    ENDM
```

IN$ MACRO string-name

```
;;
;; Read keyboard characters into the specified buffer in the
;; data segment.
;;
    PUSH  DX
    LEA    DX, string-name
    IN$-DX
    POP DX
    ENDM
```

IN$ DX MACRO

```
;;
;; Read keyboard characters into a buffer in the data segment.
;; (DS:DX) = Buffer address.
    PUSH   AX            ;; Select keyboard read option.
    MOV   AH,0AH         ;; Read keyboard.
    INT  21H
    POP   AX
      ENDM
```

DELAY MACRO minutes, seconds, hundredths

```
    LOCAL SECS, MINS, HRS, CHECK, QUIT
;;
;;  Wait for a specified interval
    PUSH   AX
    PUSH   BX
    PUSH   CX
    PUSH   DX               ;; Read current time.
    READ-TIME               ;; Copy hours into AH.
    MOV    AH, CH           ;;      minutes into AL.
    MOV    AL, CL           ;;      seconds into BH.
    MOV    BH,DH            ;;      hundredths into BL
    MOV    BL,DL
;;
;;  Add the input values to the current time to get the target
;;  time.
;;
    ADD    AL, minutes
    ADD    BH, seconds
    ADD    BL, hundredths
;;
;;  Propagate any carryover.
;;
    CMP    BL, 100
    JB     SECS
    SUB    BL, 100
    INC    BH
SECS: CMP  BH, 60
    JB     MINS
    SUB    BH, 60
    INC    AL
MINS: CMP  AL, 60
    JB     HRS
    SUB    AL, 60
    INC    AH
HRS:  CMP  AH, 24
    JNE    CHECK
    SUB    AH, AH
;;
;;Wait for interval to elapse.
;;
CHECK: READ - TIME
```

```
CLS MACRO
;;
;;  Clear the screen.
      PUSH   AX
      PUSH   BX         ;;  Save affected registers.
      PUSH   CX
      PUSH   DX
      MOV    CX,0
      MOV    DH,24      ;;  Start at row0, colmn 0.
      MOV    DL,79      ;;  End at row 24
      MOV    AH,6       ;;  and column 79.
      MOV    AL,0       ;;  Select scroll-up option.
      MOV    BH,7       ;;  Clear entire screen.
      INT    10H
      PCP    DX         ;;  Issue Video 1/0 interrput.
      POP    CX         ;;  Restore flags
      POP    BX
      POP    AX
      ENDM
```

```
CR  MACRO
;;
;; Move cursor to beginning of current line.
;;
      PUSH   CX
      PUSH   DX
      POS               ;;  Read row number into DH.
      SUB    DL, DL     ;;  Make column number 0.
      MOVE-CURSOR       ;;  Move the cursor.
      POP    DX
      POP    CX
      ENDM
```

```
CRLF  MACRO
;;
;; Move cursor to beginning of next line.
;;
      CR                ;;  Carriage return.
      LF                ;;  Line feed.
       ENDM
```

BEEP MACRO

```
;;
;; Beep the speaker for 1/2 second at a frequency of 1000 Hz
;;
    SOUND  1000, 50
    ENDM
```

MACRO string-name

```
    LOCAL FILL-BUFF, CLR-DVD, NO-MORE
;;
    ;; Convert a signed number in AX to a string in the data segment.
    ;; String must be seven bytes long. The first byte receives the
    ;; character count
    PUSH    DX
    PUSH    CX
    PUSH    BX                        ; save affected registers
    PUSH    SI
    PUSH    AX
    LEA     BX, string-name + 1  ;; BX points to first character.
    MOV     CX, 6      ;; Fill buffer with spaces
FILL- BUFF: MOV   BYTE PTR [BX], ' '
    INC     BX
    LOOP    FILL-BUFF
    MOV     SI, 10                    ;; Get ready to divide by 10
    OR      AX, AX                    ;; If value is negative,
    JNS     CLR-DIV
    NEG     AX                        ;; make it positive.
CLR-DVD: SUB   DX, DX                 ;; Clear upper half of dividend
    DIV     SI                        ;;Divide AX by 10.
    ADD     DX, '0'                   ;; Convert remainder to ASCII digit
    DEC     BX                        ;; Back up through buffer.
    MOV     [BX], DL                  ;; Store character in string.
    OR      AX, AX                    ;; Done?
    JNZ     CLR-DVD                   ;; If not, get next digit.
    POP     AX                        ;; Yes. [illegible] original value.
    OR      AX, AX                    ;; Was it negative?
    JNS     NO-MORE
    DEC     BX                        ;; Yes. Store sign.
    MOV     BYTE PTR [BX] , '-'
NO-MORE:  MOV STRING-NAME, 6          ;; Record character count
    POP     SI                        ;;Restore registers.
    POP     BX
    POP     CX
    POP     DX
          ENDM
```

```
        JA        NO-GOOD
GOOD:   CLC
        JNC       THRU
NO-GOOD: STC                           ;; If so, set Carry.
THRU:   POP  CX                        ;; Restore registers
        POP       BX
        JMP       SKIPIT
;;
;;  This subroutine performs the actual conversion.
;;
CONV-AB  PROC
        PUSH  BP                       ;; Save scratch registers
        PUSH  BX
        PUSH  SI
        MOV   BP, BX                   ;; Put pointer in BP
        SUB   BX, BX                   ;; and clear BX.
RANGE:  CMP  BYTE PTR DS: [BP], '0'    ;; If character is not
        JB    NON-DIG                  ;; a digit,
        CMP   BYTE PTR DS: [BP], '9'
        JBE   DIGIT
NON-DIG: STC                           ;; set Carry
        JC    END-CONV                 ;; and exit.
DIGIT: MOV   SI, 10                    ;; The character is a digit,
        PUSH  DX
        MUL   SI                       ;; so multiply AX by 10
        POP   DX
        MOV   BL, DS: [BP]             ;; Fetch ASCII code,
        AND   BX, OFH                  ;; save only low bits,
        ADD   AX, BX                   ;; and update result.
        JC    END-CONV                 ;; Exit if result is too Large.
        INC   BP                       ;; Otherwise increment Bp
        LOOP  RANGE                    ;; and continue
        CLC                            ;; When done, clear Carry.
END-CONV: POP SI                       ;; Restore registers.
        POP  BX
        POP  BP
        RET
CONV-AB  ENDP
SKIPIT:    NOP
        ENDM
```

MACRO string-name

```
    LOCAL  BLANKS, CHK-NEG, CHK-POS, GO-CONV, GOOD, NO-GOOD,
    THRU
    LOCAL  CONV-AB, RANGE, NON-DIG,DIGIT, END-CONV, SKIPIT
;;
;;  Convert the specified string to a signed binary number
;;  in AX. The first byte in the string must hold the
;;  character count.
;;  If the conversion is successful, CF=0 ;; otherwise CF=1.
    PUSH   BX                  ;; Save working registers.
    PUSH   CX
    SUB    AX, AX              ;; To start, rest = 0.
    SUB    CH, CH              ;; Read count into CX.
    MOV    CL, string-name
    LEA    BX, string-name +1  ;; Put first address in BX.
BLANKS: CMP  BYTE PTR [BX], '' ;; Scan past leading blanks
    JNE    CHK-NEG
    INC    BX
    LOOP BLANKS
;;
;;  Execute these instruction if the string starts with a
;;  minus sign.
;;
CHK-NEG:   CMP   BYTE PTR [BX], '-'
    JNE    CHK-POS
    INC    BX               ;; Increment the pointer.
    DEC    CX               ;; Decrement the count.
    CALL   CONV-AB          ;; Convert the string.
    JC     THRU
    CMP    AX, 32768        ;; Is the number too small?
    JA     NO-GOOD
    NEG    AX               ;; No. Complement the result.
    JS     GOOD
;;
;;  Execute these instruction if the first string character
;;  is not a minus sign
;;
CHK-POS:   CMP  BYTE   PTR [BX], '+'
    JNE  GO-CONV
    INC  BX                 ;; Increment the pointer.
    DEC  CX                 ;; Decrement the count.
GO-CONV:  CALL   CONV-AB    ;; Convert the string
    JC   THRU
    CMP     AX, 32767       ;; Is the number too large?
```

مكتبة البرامج الماكروية

التعليمات المستخدمه

monic	Description
A	ASCII Adjust for Addition
D	ASCII Adjust for Division
M	ASCII Adjust for Multiplication
S	ASCII Adjust for Subtraction
C	Add with Carry
D	Add
ID	And
LL	Call
W	Convert Byte to Word
.C	Clear Carry
.D	Clear Direction
.I	Clear Interrupt
AC	Complement Carry
AP	Compare
APB	Compare Byte (of string)
APW	Compare Word (of string)
VD	Convert Word to Double Word
AA	Decimal Adjust for Addition
AS	Decimal Adjust for Subtraction
EC	Decrement
IV	Divide
SC	Escape
LT	Halt
IV	Integer Divide
IUL	Integer Multiply
I	Input
IC	Increment
IT	Interrupt
ITO	Interrupt on Overflow
ET	Interrupt Return
	Jump on Above
E	Jump on Above or Equal
	Jump on Below
E	Jump on Below or Equal
XZ	Jump on CX Zero
	Jump on Equal
	Jump on Greater
E	Jump on Greater or Equal
.	Jump on Less
.E	Jump on Less or Equal
AP	Jump
IA	Jump on Not Above
IAE	Jump on Not Above or Equal
NB	Jump on Not Below
NBE	Jump on Not Below or Equal
NE	Jump on Not Equal
NG	Jump on Not Greater
NGE	Jump on Not Greater or Equal
NL	Jump on Not Less
NLE	Jump on Not Less or Equal
NO	Jump on Not Overflow
INP	Jump on Not Parity
INS	Jump on Not Sign
INZ	Jump on Not Zero
IO	Jump on Overflow

Mnemonic	Description
JP	Jump on Parity
JPE	Jump on Parity Even
JPO	Jump on Parity Odd
JS	Jump on Sign
JZ	Jump on Zero
LAHF	Load AH with Flags
LDS	Load Pointer into DS
LEA	Load Effective Address
LES	Load Pointer into ES
LOCK	Lock Bus
LODS	Load string
LOOP	Loop
LOOPE	Loop While Equal
LOOPNE	Loop While Not Equal
LOOPNZ	Loop While Not Zero
LOOPZ	Loop While Zero
MOV	Move
MOVS	Move string
MUL	Multiply
NEG	Negate
NOP	No operation
NOT	Not
OR	Or
OUT	Output
POP	Pop
POPF	Pop Flags
PUSH	Push
PUSHF	Push Flags
RCL	Rotate through Carry Left
RCR	Rotate through Carry Right
REP	Repeat
RET	Return
ROL	Rotate Left
ROR	Rotate Right
SAHF	Store AH into Flags
SAL	Shift Arithmetic Left
SAR	Shift Arithmetic Right
SBB	Subtract with Borrow
SCAS	Scan string
SHL	Shift Left
SHR	Shift Right
STC	Set Carry
STD	Set Direction
STI	Set Interrupt
STOS	Store string
SUB	Subtract
TEST	Test
WAIT	Wait
XCHG	Exchange
XLAT	Translate
XOR	Exclusive Or

المراجع

1. Abel, Peter
IBM PC Assembler language and Programming.
Prentic Hall, 1987.

2. Hawksley C., White N.
Assembly Language Programming On.
IBM PC.
Addison-Wesley, 1987.

3. Leventhal L.
Microcomputer Experimentation With IBM PC.
Holt, Rinehart And Winston, Inc, 1988.

4. Liu Y., Gibson G.
Microcomputer Systems: The 8086/8088 Family.
Prentical Hall, 1986.

5. Scanlon L.
IBM PC & XT Assembly Language:
A Guide For Programmers.
Prentice Hall, 1985.

٠٠١٫٦٤٢٢
برم برمجة الكمبيوتر بلغة التجميع/ تأليف زياد القاضي . . .
(وآخرون): .ـ عمان: دار المستقبل، ١٩٩٠.
(٢٤٩) ص
ر. أ. (٤٣٩/٧/١٩٩٠)
١ ـ البرمجة. ٢ ـ لغة التجميع. أ ـ زياد القاضي، مؤلف مشارك.

(تمت الفهرسة بمعرفة دائرة المكتبات والوثائق الوطنية)

صدر عن دار المستقبل للمؤلف
د . زياد عبد الكريم القاضي وزملائه
الكتب التالية :

١ – مقدمة في علم الحاسوب
٢ – معالجة النصوص واتمتة المكاتب
٣ – تحليل وتصميم نظم المعلومات المحوسبه
٤ – برمجة الكمبيوتر بيسك – فورتران
٥ – البرمجة بلغة كوبول
٦ – نظم التشغيل
٨ – تركيب البيانات
٩ – مقدمة في قواعد البيانات
١٠ – بحوث العمليات
١١ – البرمجة بلغة التجميع اسمبلي
١٢ – التصميم المنطقي ودوائر الكمبيوتر

دار المستقبل للنشر والتوزيع

المملكة الأردنية الهاشمية – عمان

هاتف ٦٣٦٣٢٧ فاكس ٦٥٨٢٦٣

ص.ب.١٨٤٢٤٨

دار
المستقبل للنشر والتوزيع
عمان ـ الاردن
ص ب ١٨٤٢٤٨ ـ هاتف ٦٣٦٣٢٧

الشركة الجديدة للطباعة والتجليد